Machine Learning Methods for Diagnosis, Prognosis and Prediction of Long-term Treatment Outcome of Major Depression

by

Zhi Nie

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved March 2017 by the
Graduate Supervisory Committee:

Jieping Ye, Co-Chair
Jingrui He, Co-Chair
Baoxin Li
Guoliang Xue
Jing Li

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

Major Depression, clinically called Major Depressive Disorder, is a mood disorder that affects about one eighth of population in US and is projected to be the second leading cause of disability in the world by the year 2020. Recent advances in biotechnology have enabled us to collect a great variety of data which could potentially offer us a deeper understanding of the disorder as well as advancing personalized medicine.

This dissertation focuses on developing methods for three different aspects of predictive analytics related to the disorder: automatic diagnosis, prognosis, and prediction of long-term treatment outcome. The data used for each task have their specific characteristics and demonstrate unique problems. Automatic diagnosis of melancholic depression is made on the basis of metabolic profiles and micro-array gene expression profiles where the presence of missing values and strong empirical correlation between the variables is not unusual. To deal with these problems, a method of generating a representative set of features is proposed. Prognosis is made on data collected from rating scales and questionnaires which consist mainly of categorical and ordinal variables and thus favor decision tree based predictive models. Decision tree models are known for the notorious problem of overfitting. A decision tree pruning method that overcomes the shortcomings of a greedy nature and reliance on heuristics inherent in traditional decision tree pruning approaches is proposed. The method is further extended to prune Gradient Boosting Decision Tree and tested on the task of prognosis of treatment outcome. Follow-up studies evaluating the long-term effect of the treatments on patients usually measure patients' depressive symptom severity monthly, resulting in the actual time of relapse upper bounded by the observed time of relapse. To resolve such uncertainty in response, a general loss function where the hypothesis could take different forms is proposed to predict the risk of relapse in situations where only an interval for time of relapse can be derived from the observed data.

Dedicated to my parents.

ACKNOWLEDGMENTS

I would like express my great gratitude toward my advisor Dr. Jieping Ye for his support, encouragement and guidance during the past five years. He has been an outstanding mentor and it has always been a great pleasure to work with him. Also, I would like give thanks to our collaborators from Johnson & Johnson including Dr. Qingqin Li and Dr. Gayle Wittenberg for some of the data resources they have been able to provide us with and a great many fruitful discussions we had during our collaboration. Furthermore, I would like to give dissertation committee co-chair Dr. Jingrui He, dissertation committee members Dr. Guoliang Xue, Dr. Baoxin Li and Dr. Jing Li for their patience and helpful advice.

Also I would like to thank previous and current members of Ye Lab including Dr. Lei Yuan, Dr. Jiayu Zhou, Dr. Shuo Xiang, Dr. Qian Sun, Dr. Rita Chattopadhyay, Dr. Sen Yang, Dr. Pinghua Gong, Dr. Binbin Lin, Dr. Zheng Wang, Dr. Kefei Liu, Dr. Chao Zhang, Dr. Jie Wang, Dr. Ming Lin, Yashu Liu, Tao Yang, Qingyang Li, with whom I had may inspiring discussions and fruitful project collaboration and from whom I had received valuable feedback.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Introduction

Major Depression, also clinically called Major Depressive Disorder (MDD), is a mood disorder that affects about one eighth of population in US and estimated 350 million people globally and is projected on track to be the second leading cause of disability in the world by the year 2020 Marcus *et al.* (2012). Although the symptoms of depression are usually manifested in depressed mood, loss of interest, sleep disturbance, mental retardation/agitation, etc., evidences from investigation in a series of disciplines ranging from genetics, neuroscience, clinical and behavioral science reveals that the origin of the disease can be traced back to the brain. These evidences shed invaluable light on mechanism of the disorder and offer basis on which possible medical intervention can be developed. Furthermore, recent advance in biotechnology has enabled collection of data of wide range of modalities, such as gene expression profiles, metabolic profiles, functional MRI etc., making it possible to gain a deeper understanding of the disorder from different perspectives by carrying out meaningful statistical analysis of these data via tools and techniques in the fast growing area of machine learning.

Unlike certain cardiovascular disease of which the symptoms are clear and not that much different from one person to another and diagnosis can be conducted with objective standards, MDD is of such heterogeneous nature that symptoms demonstrated by one patient with MDD can be drastically different from those demonstrated by

another and its diagnosis is more or less subject to subjective criteria , which complicates the study and treatment of such an affective disorder. In fact, in DSM-5, different subtypes of MDD has been defined, including melancholic depression, atypical depression, etc. Different subtypes are characterized by significantly different symptoms. For instance, melancholic depression is characterized by a loss of pleasure in most or all activities, a quality of depressed mood more pronounced than that of grief or loss, etc. Whereas atypical depression is characterized by increased appetite or weight gain, sleepiness or excessive sleep, marked fatigue or weakness. Sometimes co-occurrence of more than one subtype, namely, comorbidity might be found in a single patient.

Thus far, faced with a disorder with such heterogeneous nature, different studies have collected a variety of types of data with different objectives. In a joint investigative effort led by Johnson & Johnson and Brain Resource company, for instance, metabolic profiles, gene expression profiles and clinical data are collected on about two hundred subjects with a particular emphasis on understanding the biological basis underlying the melancholic depression which is generally believed to be a biology-based subtype of depression. In a National Institute of Mental Heath (NIMH) funded study, Sequenced Treatment Alternatives to Relieve Depression (STAR*D), which has an overall objective of determining the most effective treatment strategies and specific treatment options for patients with major depressive disorder who do not benefit adequately from initial treatment with an antidepressant, has collected clinical trial data evaluating the severity of patients' depressive symptoms and the impact of depression on their daily work and social life during both acute treatment stages and a twelve-month follow-up phase. In general, different types of data take on different types of values and each has its own peculiar characteristics which depends on the specific techniques used to collect them. For instance, the gene expression profiles

and metabolic profiles take on continuous values while the clinical trial data, usually collected through having patients or clinicians filling out certain evaluation scales, are more likely to take on categorical or ordinal values. The techniques used to collect data also play a role in determining the characteristics or even the problems that the data might exhibit. Metabolic profiles, where the metabolites are usually identified and quantified by Gas chromatograph-Mass spectrometry, entails certain amount of missing values. The levels of expression of genes which are usually measured through DNA microarray, might exhibit strong correlation among the genes sharing the same biological pathways. The variables from the data collected through questionnaires or evaluation scales which consists of items that give responders a few choices take on only a small handful of values. For instance, the data collected from QIDS which were used to measure patients' depressive symptom severity in the STAR*D study, contains variables that take on 0,1,2,3 which correspond to no symptom, slight symptom, severe symptom and extremely severe symptom, respectively. It should be noted that although these numbers are associated with different categories, the order of these values is actually meaningful. It is also interesting to observe that although the difference between any two adjacent pairs of ratings may not be the same, they are mapped to equally spaced utility values. With these different characteristics and different problems associated with each type of data, it is important that we design and apply different statistical models fit for each type of data to answer the questions we ask into these data.

1.2 Research Objectives

1.2.1 Automatic Diagnosis

Clinical diagnosis of depression is usually made by having either clinicians or patients themselves rate certain depressive symptom evaluation scale. A patient is diagnosed with depression or certain subtype of depression if the total score on that scale or total score on a corresponding sub-scale is greater than some threshold. Although depression in general may have to do with patients' personality, disposition or the problems they are confronted with in their life, some subtype of depression, such as melancholic depression, is widely considered as mainly biologically based rather than determined by personality or life circumstances. Therefore, building classification/regression models that could accurately predict whether or not melancholic depression is present in subjects based on biological data such as metabolic profiles, gene expression profiles, etc. using machine learning approaches could potentially offer us a deeper understanding the pathogenesis of the disease.

However, due to the presence of missing values in data, we either have to delete the samples with missing values or impute the missing values before we can directly apply most of classification/regression models. Furthermore, the existence of strong empirical correlation between the variables, a problem that is almost unavoidable when the number of samples is small and the number of features is huge, can lead to overfitting in most of the commonly used prediction models. In fact, there are a number of worksBühlmann *et al.* (2013) which have adopted a two-stage approach consisting of clustering or grouping the variables before pursuing fitting models. In chapter 2 of this proposal, a method for identifying the representative features from data is proposed. By adapting sparse coding into a technique for soft-clustering, the

method is capable of handling the strong empirical correlation in the data as well as the missing values in the data matrix.

1.2.2 Prognosis

In this proposal, prognosis is specifically defined as the likely treatment outcome after the patients received one or multiple stages of medical treatment with medications for each treatment stage vigorously dosed under patientsï£¡ tolerance and provided with sufficient durance. One of the studies that involves administering medications to a large demographically representative sample of depressive patients and regularly monitoring depressive status is STAR*D study which consists of a total of four treatment stages with medications provided to patients for free during each treatment stage that lasts twelve to fourteen weeks. During the acute treatment stages in the STAR*D study, depressive symptom severity were evaluated every two to three weeks. Patients that could not achieve remission or suffered from intolerable side effects of medications in one treatment stage were encouraged to proceed to the subsequent stage. Those who did achieve remission or demonstrated significant symptomatic improvement were invited to enter a 12-month naturalistic follow-up phase where assessments of patients' depressive symptom severity were made on a monthly basis.

As is mentioned in the previous subsection, most of the covariates collected in the STAR*D study are from evaluation scales or questionnaires and take on values from a few categories. One family of models that are widely held as having a natural advantage in dealing with categorical and ordinal type of variables is decision-tree based models. Specifically, in some application scenarios, it is regarded as superior to linear models due to the fact that it naturally takes into account the interaction between variables. It is also favored by data analysts when the interest lies not only in

obtaining an high predictive performance but also in pursuing the interpretability of the predictive models. However, it suffers from a major notorious problem of overfitting. Usually, some sort of post-pruning or pre-pruning techniques have to be applied in order to prevent the decision tree from overfitting and obtain a useful decision tree model. In Chapter 2, a decision tree pruning approach is proposed which basically combines the advantage of naturally modeling the interaction between the variables from the decision tree model and the advantage of less likely to overfitting of regularized linear models and formulates the decision tree pruning as a l_1 regularized linear regression with some non-convex constraint. Specifically, this decision tree pruning technique will be used to prune decision tree models for predicting the treatment outcome by the end of the level two treatment based on covariates collected from the enrollment and up to week two of the level one treatment.

1.2.3 Prediction of Long-Term Treatment Outcome

Another problem concerning the treatment of MDD is the potential risk of relapse after patients initially achieve remission after one or several stages of treatment with medications. Although there is a substantial risk of relapse among the remitted patients, follow-up studies tracking those patients also observed an interesting phenomenon that the longer a patient survives, the less likely that a patient will eventually relapse. In other words, the risk of relapse is inversely proportional to the length of time that effect of the treatment lasts. One promising approach to predicting patients' risk of relapse is survival analysis which is typified by Cox model. However, most of the follow-up studies that track patients' depressive status only evaluate patients' depressive symptoms on a monthly basis, which introduces uncertainty in the responses of the uncensored cases, making it difficult to be handled by the traditional survival analysis models such as Cox model. To employ those mod-

els, for those patients that eventually relapsed in the follow-up study, a time point between the last time point when the patient is observed normal and the time when relapse is observed has to be picked and used as observed response to build the model. To model such uncertainty, we proposed a truncated l_1 loss in which the hypothesis that could take of form of (1) ensemble of decision tress; (2) linear combination of covariates. Furthermore, the linear model is extended to deal with the case where the covariates from more than one treatment stage is involved.

Chapter 2

MELANCHOLIC DEPRESSION PREDICTION BY IDENTIFYING REPRESENTATIVE FEATURES IN METABOLIC AND MICROARRAY PROFILES WITH MISSING VALUES

2.1 Introduction

Understanding the fundamental biology underlying melancholic depression is a very challenging problem of great clinical importance for researchers from medical and psychiatric research communities. Unlike some other subtypes of depression, melancholic depression is described as "mainly biologically based rather than determined by personality or life circumstances"McGrath *et al.* (2008), which motivates researchers to discover biological evidence of the disease. Research with regard to this aspect has made progress in recent years. For instance, it has been shown that an elevated level of concentration of certain metabolites in plasma is found among the depressive patients with melancholia Mazure *et al.* (1987). More recently, Gabbay *et al.* Gabbay *et al.* (2010) pointed out the significance of kynurenine pathway in adolescent depression with melancholic features through comparing adolescents with melancholic depression with non-melancholic depression and healthy adolescents. Also, recently through gene ontology and pathway analyses, certain biological functions of differentially expressed mRNAs were identified as related to fundamental metabolic processes and brain disordersLiu *et al.* (2014).

On the other hand, recent advances in biotechnologies have made it possible to detect a large number of metabolites from human tissue extract. Meanwhile, the microarray technology has taken us from being able to analyze the biological functions of

only a few related genes or proteins at one time to the place where global investigation of cellular activities is possible Hoheisel (2006). With data on such a large scale available, one promising approach that can potentially offer us a deeper understanding of collective impact of numerous factors involved in the pathogenesis of melancholic depression and its prospective treatments is to build predictive models based on all of the information available using machine learning approaches. However, the "curse of dimensionality" due to the fact that the number of variables of interest far exceeds the number of samples available renders most of traditional classification/regression algorithms less effective in this setting. Furthermore, strong empirical correlations between the variables, especially in the case of microarray data where there is high degree of linear dependence between expression measures of a group of genes sharing the same biological pathwaysSegal *et al.* (2003), tremendously limit the prediction performance of traditional machine learning. Another major issue with data collected on a large scale is the presence of missing values, which is ubiquitous in biomedical applications.

Most of the existing methods are designed to deal with either the problem of strong empirical correlations between the variables or the problem of missing values. For instance, Bühlmann *et al.* (2013) recently proposed a bottom-up agglomerative clustering algorithm to deal with correlations between the variables, but their method cannot be readily used in the context of missing values. As for the issue of missing values, as is pointed out in by Thung *et al.*Thung *et al.* (2014), basically there are two approaches to dealing with missing data. We can either discard the samples with missing values or impute the missing data. The shortcoming of the first approach is obvious. It does not make full use of available information. The second approach, imputation of missing data, generally involves certain assumptions about the missing pattern of the data which are not satisfied in applications. The most classic impu-

tation technique, EM, for example, can work well only when data is sampled from Gaussian distribution and missing-at-random (MAR) assumption is satisfied.

We hypothesize that the problem of missing values might potentially benefit from the correlations between the variables; for example, a variable with missing values could borrow information from its correlated variables. However, simply imputing the missing values of a variable by exploiting information from its correlated variables still leaves the problem of empirical correlations between the variables unsolved. Therefore, instead of discarding the incomplete samples or imputing the missing values, we attempt to generate a compressed set of representative features for all the samples from the data with a group of correlated variables represented by one or a few features. We can demonstrate that sparse coding, which has been shown to be very effective in object recognition and image denoising applicationsYang *et al.* (2009); Lu *et al.* (2011), is desirable for such a task. Specifically, we apply sparse coding in such a way that the learned dictionary corresponds to a set of representative features and each variable is represented as a sparse combination of these features. Furthermore, we develop an efficient algorithm to solve the proposed sparse coding formulation to deal with missing values. We argue that with all the parameters being equal, the statistical properties (e.g. the distribution of positive samples and negative samples) of the features generated from the design matrix with missing values will not be different from those of the features we would obtain had all the missing entries been known when the missing ratio is within certain range. Note that in using the compressed set of the features generated from sparse coding to represent the original design matrix, we only assume that there exists empirical correlation among some of the variables (which is dealt with through the sparsity in the model).

We apply the proposed algorithm to datasets of metabolic and microarray profiles collected from a group of subjects consisting of both patients with melancholic

depression and healthy controls. Results from our experiments revealed that features obtained from our method significantly outperform those generated from several baseline methods based on traditional clustering methods and standard data imputation techniques. In particular, in comparison with our baseline methods, the representative features learned by the proposed method achieve much better performance in predicting the disease status of the subjects with melancholic depression on both datasets. In addition, on the dataset of metabolic profiles, we found that most of the known metabolites within each cluster are biologically relevant. These results demonstrate the promise of the proposed method for learning from incomplete and high-dimensional biomedical data.

The contribution of this chapter is manifested in the following aspects:

- The classification performance achieved by applying a simple linear svm to our representative set of features is significantly better than

- On datasets of metabolite and microarray collected from a group of subjects comprising both patients with melancholic depression and healthy controls, features obtained from our method significantly outperform those generated from traditional clustering and data imputation techniques in tasks of classification. In particular, on both datasets, we found that in comparison with those traditional clustering algorithms, feature sets yielded by sparse coding give rise to significantly improved sensitivity scores, suggesting that learned features allow prediction with high accuracy of disease status in those who are diagnosed with melancholic depression.

- A closer look at the cluster structure reflected by the sparse coefficient matrix obtained from sparse coding gives us new insights into the groups of correlated variables. Interestingly, on the dataset of metabolic profiles, we found that most

of the known metabolites within each cluster are biologically relavant according to the ontology.

The rest of the chapter is organized as follows. In section 2.2, we formulate the sparse coding problem in the presence of missing values. In section 2.3, we describe the dataset used in the analysis and present experimental results.

2.2 Learning Representative Features via Sparse Coding

In this section, we present our sparse coding formulation to learn a compressed representative set of features such that the observations from all the samples on each variable can be represented as a sparse linear combination of these learned features. The proposed formulation can naturally deal with missing values.

Suppose we are given a dataset of m samples and their observations on n variables with missing values which we denote as $\mathcal{X} = \{(x_1, \Omega_1), \cdots, (x_n, \Omega_n)\}$. Each x_i $(1 \leq i \leq n)$ is an m dimensional column vector representing measurements of all the samples on the i-th variable (e.g., concentrations of the i-th metabolite or measurements of the i-th gene expression), and Ω_i is an ordered set of integers ranging from 1 to m including the indices of samples whose measurements on the i-th variable are observed. If there is no missing value in x_i, then Ω_i includes all integers between 1 and m. Our goal is to use sparse coding to learn a set of k representative features such that each variable x_i can be well represented by a sparse combination of these k features. In the presence of missing values, the sparse coding problem can be formulated as the following optimization problem:

$$\begin{aligned} &\min_{\mathcal{D}, z_1, \cdots, z_n} \sum_{i=1}^{n} \frac{1}{2} \|\mathcal{P}_{\Omega_i}(\mathcal{D}z_i - x_i)\|_2^2 + \lambda \|z_i\|_1 \\ &s.t. \qquad \|\mathcal{D}_{\cdot j}\|_2 \leq 1; 1 \leq j \leq k, \end{aligned} \tag{2.1}$$

where z_i represents sparse combination coefficients (also called sparse code) for x_i,

and $\mathcal{D} \in \mathbb{R}^{m \times k}$, the dictionary or codebook, represents the learned set of features with its j-th column denoted as $D_{\cdot j}$. $\mathcal{P}_{\Omega_i}(\cdot)$ projects a matrix into its submatrix consisting of rows indexed by Ω_i. The cardinality of Ω_i is denoted by m_i. Minimization of the first term in (2.1) leads to a feature set $\mathcal{D}$ such that the observed entries of each variable can be well represented by the features in $\mathcal{D}$, thus ensuring variables with similar combination coefficients are correlated. Minimization of the second term in (2.1) induces sparsity on combination coefficients of each variable, enforcing each variable to be represented by only a small subset of features in $\mathcal{D}$. λ controls the sparsity of each z_i. The larger the λ is, the sparser each z_i will be. With a proper λ, minimization of these two terms combined will yield a feature set $\mathcal{D}$ such that the observed part of each variable can be well represented by a small subset of features from $\mathcal{D}$.

Although the problem in (1) is convex with respect to either z_i or $\mathcal{D}$, it is not jointly convex. Thus, it is difficult to obtain a globally optimal solution. Most algorithms solving the sparse coding problem alternate the step of optimizing over z_i with a fixed $\mathcal{D}$ and the step of optimizing over $\mathcal{D}$ with a fixed z_i Mairal *et al.* (2010). In this chapter, we extend the framework proposed by Lin *et al.* Lin *et al.* (2014), which applies to data without missing values, to solve sparse coding with missing values in data matrix. The detailed description of the algorithm to solve the above problem is presented in Algorithm 1.

With a fixed $\mathcal{D}$, updating z_i amounts to solving a LassoTibshirani (1994) problem which can be formulated as follows:

$$\min_{z_i} f_{\mathcal{D}}(z_i) \equiv \frac{1}{2}\|\mathcal{P}_{\Omega_i}(\mathcal{D}z_i - x_i)\|_2^2 + \lambda\|z_i\|_1. \tag{2.2}$$

Suppose that we iteratively update the sparse code of each variable for T epochs, The total number of Lasso problems involved is Tn. Even with state-of-the-art solvers,

the total cost of solving so many Lasso problems is prohibitive, particularly in the case of the microarray data where there are usually at least tens of thousands of genes involved. In our algorithm, we adopt the strategy of solving the Lasso problem incrementally by updating only the support of z_i for a few times via coordinate descent with a warm start. This strategy has proved to be computationally efficient in practice while still yielding competitive performance. The description of coordinate descent is given in Algorithm 2.2.

Each row of the learned dictionary $\mathcal{D}$ represents a sample and each column represents a feature. The classification will be carried out on $\mathcal{D}$.

In the algorithm, each time we pick one element, say the j-th element z_{ij} ($1 \leq j \leq k$), to update with all the other coordinates fixed. Under this circumstance, (2.2) can be converted to a problem with closed form solution. Let $z_i = z_i^s$ before z_{ij} is updated. Let $z_i = z_i^{s+1}$ after z_{ij} is updated. Let $\overline{z}_{ij} = [z_{i,1}, \cdots, z_{i,j-1}, z_{i,j+1}, \cdots, z_{i,k}]^T$, $\overline{\mathcal{D}}_{\cdot j} = [\mathcal{D}_{\cdot 1}, \cdots, \mathcal{D}_{\cdot j-1}, \mathcal{D}_{\cdot j+1}, \cdots, \mathcal{D}_{\cdot k}]$. Apparently, $\overline{z}_{ij}^{s+1} = \overline{z}_{ij}^s$, z_{ij}^{s+1} is the only unknown variable. Plugging z_i^{s+1} into $f_{\mathcal{D}}$, we have

$$\begin{aligned} f_{\mathcal{D}}(z_i^{s+1}) &= \frac{1}{2}\|\mathcal{P}_{\Omega_i}(x_i - \overline{\mathcal{D}}_{\cdot j}\overline{z}_{ij}^{s+1})\|_2^2 - \left(\mathcal{P}_{\Omega_i}(x_i - \overline{\mathcal{D}}_{\cdot j}\overline{z}_{ij}^{s+1})\right)^T \mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j}) z_{ij}^{s+1} \\ &\quad + \frac{1}{2}\|\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})\|_2^2 (z_{ij}^{s+1})^2 + \lambda |z_{ij}^{s+1}| + \lambda \|\overline{z}_{ij}^{s+1}\|_1 \\ &= \frac{1}{2}\|\mathcal{P}_{\Omega_i}(x_i - \overline{\mathcal{D}}_{\cdot j}\overline{z}_{ij}^{s})\|_2^2 - \left(\mathcal{P}_{\Omega_i}(x_i - \overline{\mathcal{D}}_{\cdot j}\overline{z}_{ij}^{s})\right)^T \mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j}) z_{ij}^{s+1} \\ &\quad + \frac{1}{2}\|\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})\|_2^2 (z_{ij}^{s+1})^2 + \lambda |z_{ij}^{s+1}| + \lambda \|\overline{z}_{ij}^{s}\|_1. \end{aligned}$$

Following the practice from Mairal *et al.* Mairal *et al.* (2010) and Lin *et al.* Lin *et al.* (2014), we enforce $\|\mathcal{D}_{\cdot j}\|_2 = 1$. By setting $\partial f_{\mathcal{D}}(z_i^{s+1})/\partial z_{ij}^{s+1} = 0$, we have

$$-(\mathcal{P}_{\Omega_i}(x_i - \overline{\mathcal{D}}_{\cdot j}\overline{z}_{ij}^{s}))^T \mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j}) + \|\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})\|_2^2 (z_{ij}^{s+1}) + \lambda \operatorname{sign}(z_{ij}^{s+1}) = 0.$$

Adding and subtracting $\|\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})\|_2^2 z_{ij}^s$, we have

$$-(\mathcal{P}_{\Omega_i}(x_i - \mathcal{D} z_i^s))^T \mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j}) + \|\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})\|_2^2 (z_{ij}^{s+1}) - \|\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})\|_2^2 (z_{ij}^{s}) + \lambda \operatorname{sign}(z_{ij}^{s+1}) = 0.$$

This equation has a closed form solution which is given by

$$z_{ij}^{s+1} = S_a\Big(\frac{(\mathcal{P}_{\Omega_i}(x_i - \mathcal{D}z_i^s))^T\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})}{\|\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})\|_2^2} + z_{ij}^s\Big).$$

where S is the shrinkage operator defined by $S_\alpha(x) = (|x| - \alpha)_+ \operatorname{sign}(x), x, \alpha \in \mathbb{R}$ and $a = \lambda/\|\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})\|_2^2$. Note that although x_i may have missing values, z_i does not contain missing values. Only the rows of $\mathcal{D}$ corresponding to the rows of x_i where values are observed are used to update z_i. It is worth emphasizing that we only update all the coordinates of z_i in the first iteration due to the fact that the dictionary has changed since z_i was updated last time. For iterations afterwards, only the support of z_i is updated.

With a fixed z_i, we only use the newly updated z_i and the corresponding x_i to update $\mathcal{D}$ using gradient descent. The problem can be formulated as follows:

$$\min_{\mathcal{D}} g_{z_i}(\mathcal{D}) \equiv \frac{1}{2}\|\mathcal{P}_{\Omega_i}(x_i - \mathcal{D}z_i)\|_2^2 \quad s.t.\ \|\mathcal{D}_{\cdot j}\|_2 \leq 1,\ 1 \leq j \leq k. \tag{2.3}$$

The gradient of g_{z_i} with respect to $\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})$ is $\nabla_{\mathcal{P}_{\Omega_i}(\mathcal{D}_{\cdot j})} g_{z_i} = \mathcal{P}_{\Omega_i}(\mathcal{D}z_i - x_i)z_{ij}$. Note that only the columns of $\mathcal{D}$ corresponding to the support of z_i need to be updated. As to the learning rate, following the practice of Mairal *et al.* (2010) and Lin *et al.* (2014), we set the learning rate to be $1/\mathcal{H}[j, j]$ where $\mathcal{H}$ is initialized to be zero and accumulates $z_i z_i^T$. Finally, $\mathcal{D}_{\cdot j}$ is normalized to be within the unit ball.

The basic idea of updating the dictionary $\mathcal{D}$ in the presence of missing values is illustrated in Figure 2.1.

Note that most existing works apply sparse coding on signals, e.g., images to learn a representation of each signal. One novel aspect of the proposed framework is that we apply sparse coding to learn a sparse representation for each variable and use the dictionary $\mathcal{D}$ as features where each row represents a sample and each column represents a feature. In addition, most sparse coding formulations assume that

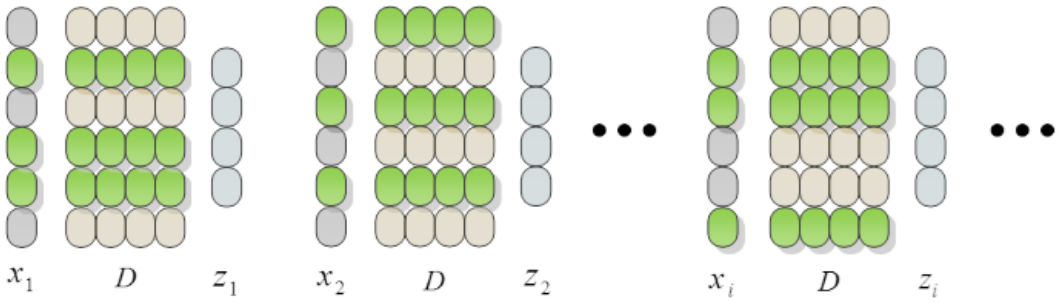


Figure 2.1: Updating the dictionary in the presence of missing values. In the process of updating dictionary, rows (yellow ovals) corresponding to the missing entries of a variable (grey ovals) are left aside while only the rows corresponding to the observed entries of a variable (green ovals) are changed. However, when each column of the dictionary is normalized to be within a unit ball, the whole dictionary is updated.

the data is complete, while our proposed framework can naturally deal with missing values in the data. From the perspective of clustering, different from those traditional clustering algorithms such as Kmeans which assign each data point to one cluster, sparse coding can be considered as a soft version of clustering in that it allows a point to belong to different clusters at the same time, depending on the number of non-zero elements in its sparse representation vector. Such flexibility is desirable in many applications, since some data points may be close to multiple clusters. The sparse representation of a data point may be a zero vector, especially when regularization parameter λ in the algorithm is set to be a large value. In this case, the data point can be regarded as an outlier or a noisy point.

2.3 Data and Experiments

The datasets used in our analysis were collected from a study initiated by Brain Resource Company (BRC) and Johnson & Johnson Pharmaceutical Research & Development, L.L.C(J&J PRD). The overall objective of the study is to identify the

best molecular profiles, cognitive and psychophysiological biomarkers in people with depression. In the study, about 100 depressive subjects evenly distributed in gender and age as well as an equal number of matched healthy controls are recruited nationwide by BRC in Australia. All the subjects that are included in the study have been screened to satisfy certain criteria on Hamilton Depression Rating Scale (HAM-D) score, CORE scoreParker and Hadzi-Pavlovic (1996), toxicology tests, and so on. Part of the study is dedicated mainly to collecting the following information from all the participants : a) Personal medical history; b) Cognition; c) Electrical brain-body function (EBBF); d) Brain structure (e.g. structural MRI, functional MRI); e) Molecular profiles (which includes metabolite, microarray, protein and transcripts profiles). However, not all the subjects have all five blocks of information or all sub-categories of one type of information recorded due to a variety of reasons such as participant dropout, failure of quality control, long storage time, etc. In our analysis, we use metabolite and microarray data from the molecular profiles to demonstrate the effectiveness of the proposed algorithm in dealing with correlated variables as well as the significant discriminative power of the resulting compressed set of features. As for the target, we are interested in melancholic depression. The decision of whether or not a subject is diagnosed with melancholic depression is made on the basis of pyschomotor findings in the CORE scale which consists of 18 items measuring a subject's interactiveness, motor agitation, etc. The score of each item ranges from 0 (no symptom) to 3 (severe symptom). A subject will be labeled as melancholically depressive if he or she has a total score on CORE over 8.

2.3.1 Analysis on Metabolic Profiles

Data Preprocessing

During the stage of medical screening, a sample of 20ml of plasma was obtained from each of the participants by BRC and was later on sent to J&J PRD where the molecular profiling analysis was carried out. Based on Gas chromatography-mass spectrometry (GS-MS) and Liquid chromatography-mass spectrometry (LC-MS/MS), 272 peaks were acquired with 160 of them being known metabolites and the rest unknown. Considering the fact that concentrations of metabolites change with the increase of storage time, we removed all the samples stored for over 200 days and performed a linear regression of concentration with storage time at the temperature of -20 degrees centigrade on the remaining samples to control for the confounding effects caused by storage time. Also, over 40 metabolites whose concentrations were detected to be highly correlated with storage time were excluded from our analysis. After the pre-processing, we are left with 118 samples and 228 metabolites in total. Among all the 118 samples, 21 were diagnosed with melancholic depression and 97 were healthy controls. About 1.27% of all the entries in the data matrix are missing.

The method of sparse coding proposed in this paper can deal with missing entries in the data matrix. To demonstrate the capability of our method to generate a compressed discriminative set of features even under the presence of missing values, we also include several baseline methods for comparison, which impute the missing entries using some standard missing value imputation techniques including: 1) HalfMin: Impute the missing entries on each variable by filling in half of the minimum of the observed values on that variable; 2) KNN: Find the k nearest neighbors of the variable with missing values based on observed part and assign the missing values to be a weighted combination of its nearest neighbors with the weight determined by the

inverse of the Euclidean distance between the variable concerned and the neighbor; 3) Expectation Maximization (EM): Assuming that the underlying distribution of the samples follows a mixture of Gaussian distribution, it iterates between updating the posterior probability of each of the data points and updating the mean, covariance matrix and mixing coefficient of each Gaussian component and filling in the missing entries with conditional expectation given the observed part; 4) Singular value decomposition (SVD): Assuming that there is an inherent low-rank structure in the data, it fills in the missing entries with the values obtained from the low-rank approximation of the data.

Before further data analysis, each variable was normalized to have zero mean and unit standard deviation. In the case of variables with missing values, we simply omitted the missing values when computing the mean and standard deviation.

Classification

With the ratio of the number melancholic depressive subjects to the number of healthy controls being almost 1 to 5, the dataset is extremely imbalanced. Direct application of traditional classification methods like Support Vector Machine (SVM) in this situation would severely biased the classifier toward majority class. Drawing on the experience from Dubey *et al.* (2014), we implemented a scheme which combines the techniques of data under-sampling and model ensemble methods to deal with the issue of data imbalance.

In this scheme, samples from each of the two classes were randomly partitioned into 10 folds of (approximately) equal size. One fold from both classes were set aside for testing and the rest were used as training set. During the training stage, we used all the samples from the minority class and randomly subsampled with replacement the same amount of samples from the majority class to build a classifier. The process

of subsampling was repeated p times (in our experiments, we choose $p = 30$) so that p different classifiers will be built on the same training set. Each of the p classifiers will give a prediction of the label of each testing sample. In the ensemble stage, predictions from different classifiers were combined in different ways. In our experiments we adopted two strategies to combine the predictions from different classifiers. The first strategy counts the number of times that a given testing sample is predicted positive and the number of times the sample is predicted negative. The final label of the sample is given by the majority of the votes. If there is a tie, then we randomly assign the testing sample to one of the two classes. The second strategy weights the prediction of each of the classifier with its confidence accompanying the prediction. The final predicted label is determined by the sign of the confidence weighted sum of prediction from each of the p classifiers. Each of the 10 folds from both classes is used as the testing fold once so that each of the samples is used as testing sample exactly once. We regard it as a convention throughout the paper that the positive class consists of subjects with melancholic depression and the negative class consists of healthy controls. The basic classifiers we used in the paper include SVM with linear kernel and Random Forest (RF). We used the following four measures to evaluate the performance of ensemble of classifiers: accuracy, sensitivity, specificity and area under curve (AUC).

In our experiments, we compared classification performance on features yielded from our method (SC, in abbreviation) with those generated by different data imputation and clustering methods. We tried different initializations, different values of K (number of keywords in the dictionary or number of clusters) and different values of λ (regularization parameter) on our method, and tried different initializations and different values of K on Kmeans and hierarchical clustering.

Table 2.1: Classification performance on metabolic profiles

	RF with majority vote			
Method	Accuracy	Sensitivity	Specificity	AUC
HalfMin	0.7624	0.6333	0.7922	0.7128
EM	0.7367	0.5833	0.7711	0.6772
KNN	0.7624	0.6833	0.7822	0.7328
SVD	0.7450	0.5833	0.7811	0.6822
HC	0.7540	0.8000	0.7489	0.7744
Kmeans	0.7778	0.7167	0.7922	0.7544
SC	**0.8315**	**0.8333**	**0.8344**	**0.8339**
	RF with weighted vote			
Method	Accuracy	Sensitivity	Specificity	AUC
HalfMin	0.7624	0.6333	0.7922	0.7128
EM	0.7290	0.5833	0.7611	0.6722
KNN	0.7624	0.6333	0.7922	0.7128
SVD	0.7547	0.6333	0.7822	0.7078
HC	0.7214	0.6500	0.7411	0.6956
Kmeans	0.7861	0.7167	0.8022	0.7594
SC	**0.8315**	**0.8333**	**0.8344**	**0.8339**

The classification performance is reported in Table 2.1. Due to the fact that SVM generally performed worse than RF on this dataset, we only report the classification performance by RF. In using KNN for data imputation, we tried a range of values for k and report the results from $k = 3$ since it gives the best performance. Also for Kmeans and hierarchical clustering, we imputed the raw design matrix using KNN

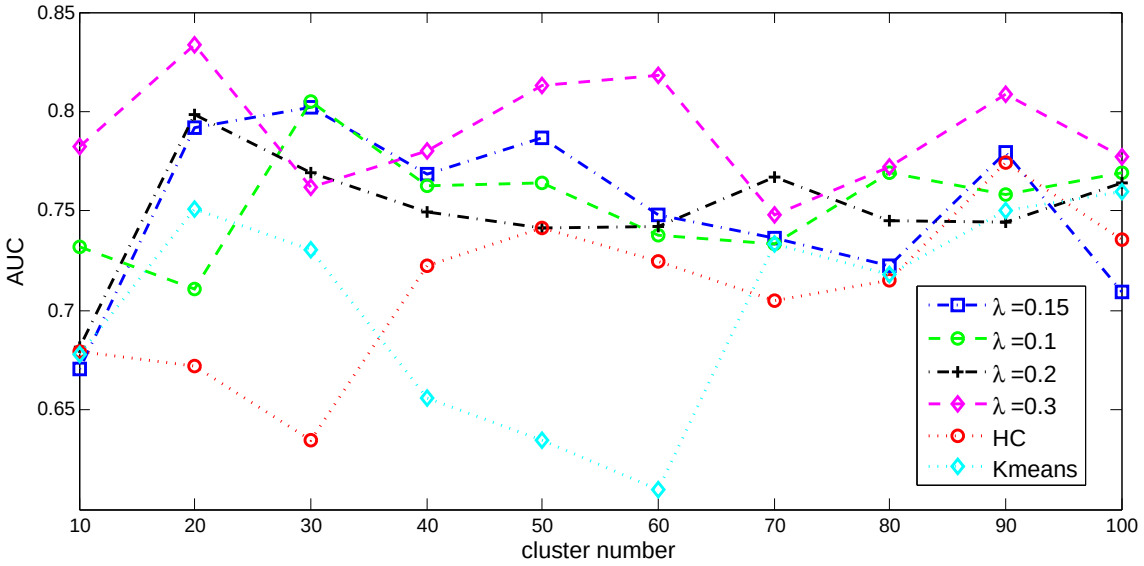


Figure 2.2: Changes in AUC score with varying number of clusters for sparse coding with different regularization parameters, Kmeans and hierarchical clustering.

before we applied these clustering techniques as KNN gave the best classification performance among all data imputation techniques. All the three clustering methods share one parameter, K, which we varied between 10 and 100 with a step size of 10. For sparse coding, there is an extra parameter λ which we set to be 0.1, 0.15, 0.2 and 0.3 in our experiments.

Fig. 1 shows how the AUC score changes when we ran classification on features generated by different clustering algorithms for different values of K. We can see that although the classification performance fluctuates with a growing number of clusters for all the clustering algorithms, sparse coding generally yields feature sets with stronger discriminative power when λ is set to be 0.3. This implies that there are indeed several clusters of metabolites in our dataset since a larger λ tends to drive the combination coefficient for each metabolite to be sparse and several metabolites are potentially outliers.

It is also of great interest to explore the groups of metabolites that are clustered together by looking into the coefficient matrix Z. The metabolites clustered into the ith group, which is represented by the ith column of $\mathcal{D}$, are those metabolites corresponding to the nonzero entries of the ith row of matrix Z. We looked into the most discriminative features (measured in terms of their p values) in the feature set $\mathcal{D}$ on which the best classification performance is achieved and their corresponding rows in the coefficient matrix Z. The most discriminative feature, which has a p-value of 2.93×10^{-16}, corresponds to six metabolites with four of them being unknown, one of them belonging to the general category of "Complex lipids, fatty acids and related" and one of them belonging to the general category of "Amino acids and related". The second most discriminative feature which has a p-value of 6.64×10^{-16} corresponds to twenty-four metabolites, with sixteen of them falling in the category "Amino acids and related", two of them belonging to the category "Nucleobases and related", five of them being unknown and one of them belonging to the category "Hormones, signal substances and related". The third most discriminative feature which has a p-value of 7.92×10^{-16} corresponds to a group of nineteen metabolites, with nine of them falling into the category "Complex lipids, fatty acids and related", six of them unknown, two of them being "unknown lipid" and one of them being "Vitamins, cofactors and related". From these, we can see that sparse coding does produce meaningful clusters in that most of the known metabolites assigned into the same cluster belong to related categories.

2.3.2 Analysis on Microarray Profiles

Microarray is another modality of data collected on the subjects. This dataset included information on 54675 transcripts from 123 subjects with 28 of them being labeled as positive (with melancholic depression) and 95 of them being labeled as

negative (normal control). The same framework introduced in 2.3.1 to deal with extreme imbalance between two classes was also used on this dataset. Among the 54675 transcripts, a list of 2261 transcripts was identified as related to depression and a list of 3297 transcripts was identified as related to immune system. We ran classification on all three sets of data. There is no missing value in this dataset.

We also compared the classification performance on features generated by sparse coding, Kmeans and hierarchical clustering. For all the clustering methods, we set the number of clusters to be 100, 200, 500, 1000. For sparse coding, we used the same set of values for λ that were used on the metabolic profile. We report the best classification performance on features generated by sparse coding on each λ over all the K values and best classification performance on features generated by Kmeans and hierarchical clustering over all the Ks. we only report the classification performance by SVM since SVM generally outperforms Random Forest on this dataset.

It is evident from Table 2.2, Table 2.3, Table 2.4 that the feature sets yielded by sparse coding have superior discriminative power than those generated by traditional clustering methods and raw data across all these three sets of data. In particular, feature sets obtained through sparse coding give rise to significantly improved sensitivity in classification performance, implying that it allows prediction with high accuracy of disease status in those who are diagnosed with melancholic depression. Overall, the feature sets given by sparse coding produce the best performance when $K = 500$. However, as shown in the results, a proper choice of λ is important as well.

2.4 Conclusion and Future Work

In this chapter, we propose a method to learn a compressed set of representative features through an adapted version of sparse coding which is capable of simultaneously clustering variables with strong empirical correlation and dealing with the

Table 2.2: Classification performance on all genes

	SVM with majority vote			
Method	Accuracy	Sensitivity	Specificity	AUC
Raw data	0.6411	0.6167	**0.6400**	0.6283
KM	0.6333	0.6833	0.6200	0.6517
HC	0.6007	0.6333	0.5878	0.6106
SC(λ=0.1)	0.6172	**0.7833**	0.5656	0.6744
SC(λ=0.15)	0.6578	0.7000	0.6389	0.6694
SC(λ=0.2)	**0.6668**	0.7667	**0.6400**	**0.7033**
SC(λ=0.3)	0.6578	0.7500	0.6289	0.6894
	SVM with weighted vote			
Method	Accuracy	Sensitivity	Specificity	AUC
Raw data	0.6090	0.6167	0.5978	0.6072
KM	0.6172	0.7167	0.5878	0.6522
HC	0.6167	0.6333	0.6089	0.6211
SC(λ=0.1)	0.6019	**0.7833**	0.5456	0.6644
SC(λ=0.15)	0.6578	0.7500	0.6278	0.6889
SC(λ=0.2)	0.6411	0.7667	0.6067	0.6867
SC(λ=0.3)	**0.6744**	0.7500	**0.6511**	**0.7006**

missing values in the design matrix. We apply the proposed method on datasets of metabolic and microarray profiles collected from a group of subjects consisting of patients with melancholic depression and healthy controls. Results show that our method can not only produce meaningful clusters of variables, but also generate a set of representative features which demonstrate superior discriminative power than

those generated by traditional clustering and data imputation techniques. In particular, on both datasets, we found that in comparison with those traditional clustering algorithms, feature sets yielded by sparse coding give rise to significantly improved sensitivity scores, suggesting that learned features allow prediction with high accuracy of disease status in those who are diagnosed with melancholic depression.

One interesting future direction is to extend the current method to deal with data with multiple modalities and block-wise missing patterns (i.e., one sample may lack observations on one ore more modalities). Simply concatenating different types of data is not appropriate in this situation since there is a high risk that pseudo-correlation may be detected between variables belonging to different data types which are not really related possibly due to a limited number of observations available on these variables. One direction is to use sparse coding to simultaneously learn a group of features shared by all data types and individual features specific to each data type.

Algorithm 1 Stochastic Coordinate Coding with Missing Values

Initialization:

Samples $X = \{x_1, x_2, \ldots, x_n\}$, missing indices $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_n\}$, $\lambda \in \mathbb{R}$, initial dictionary $\mathcal{D}^0 \in \mathbb{R}^{m \times k}$, initial combination coeficients $Z = \{z_1, z_2, \ldots, , z_n\}$, number of iterations in Coordinate Descent $l \in \mathbb{N}$, number of iterations T .

$\mathcal{H} \in \mathbb{R}^{k \times k} \leftarrow 0$

for $t = 1$ to T **do**

  for $i = 1$ to n **do**

    Update coeficients via one or a few steps of coordinate descent

    $z_i \leftarrow \arg\min_z \mathbf{CD}(\mathcal{P}_{\Omega_i}(D^{i-1}), \mathcal{P}_{\Omega_i}(x_i), z_i, \lambda, l)$.

    Update Hessian matrix

    $\mathcal{H} = \mathcal{H} + z_i z_i^T$,

    Update the dictionary $\mathcal{D}^{i-1}$ colum by column

    for $j \in \{t | 1 \leq t \leq k, t \in \mathbb{N}, z_i(t) \neq 0\}$ **do**

      $u_j = \mathcal{P}_{\Omega_i}(\mathcal{D}^{i-1}_{\cdot,j}) - \frac{1}{\mathcal{H}[j,j]} z_i(j) * \mathcal{P}_{\Omega_i}(\mathcal{D}^{i-1} * z_i - x_i)$.

      $\mathcal{P}_{\Omega_i}(\mathcal{D}^{i}_{\cdot,j}) \leftarrow u_j$.

      $\mathcal{D}_{\cdot j} \leftarrow \frac{1}{\max\{\|\mathcal{D}_{\cdot j}\|_2, 1\}} \mathcal{D}_{\cdot j}$.

    end for

  end for

  $\mathcal{D}^0 \leftarrow \mathcal{D}^n$

end for

Output: $\mathcal{D}^n$.

Algorithm 2 Coordinate Descent

Initialization:

$\mathcal{D} \in \mathbb{R}^{m_i \times k}, x \in \mathbb{R}^{m_i}, z^0 \in \mathbb{R}^k, \lambda \in \mathbb{R}, l \in \mathbb{N}$ (number of iterations)

$s \leftarrow 0$.

for $j = 1$ to k **do**

  $z^{s+1} \leftarrow z^s$.

  $\lambda_j \leftarrow \lambda / \|\mathcal{D}_{\cdot j}\|_2^2$.

  $z^{s+1}(j) \leftarrow S_{\lambda_j}(\frac{(x - \mathcal{D}z^s)^T \mathcal{D}_{\cdot j} + z^s(j))}{\|\mathcal{D}_{\cdot j}\|_2^2})$.

  $\mathbf{b} = \mathcal{D}^T(x - \mathcal{D}z^{i-1}) + z^{i-1}$

  $z^i = S_\lambda(\mathbf{b})$

  $s \leftarrow s + 1$.

end for

for $t = 2$ to l **do**

  $\mathbf{S} = \{i | i \in \mathbb{N}, 1 \leq i \leq k, z^s(i) \neq 0\}$.

  for $j \in \mathbf{S}$ **do**

    $z^{s+1} \leftarrow z^s$.

    $\lambda_j \leftarrow \lambda / \|\mathcal{D}_{\cdot j}\|_2^2$.

    $z^{s+1}(j) \leftarrow S_{\lambda_j}(\frac{(x - \mathcal{D}z^s)^T \mathcal{D}_{\cdot j} + z^s(j))}{\|\mathcal{D}_{\cdot j}\|_2^2})$.

    $\mathbf{b} = \mathcal{D}^T(x - \mathcal{D}z^{i-1}) + z^{i-1}$

    $z^i = S_\lambda(\mathbf{b})$

    $s \leftarrow s + 1$.

  end for

end for

Output: z^s .

Table 2.3: Classification performance on genes related to depression

	SVM with majority vote			
Method	Accuracy	Sensitivity	Specificity	AUC
Raw data	0.6822	0.5833	0.7056	0.6444
KM	0.6597	0.6333	0.6633	0.6483
HC	0.6681	0.6167	0.6756	0.6461
SC(λ=0.1)	0.7245	**0.7833**	0.7044	**0.7439**
SC(λ=0.15)	**0.7573**	0.7000	**0.7689**	0.7344
SC(λ=0.2)	0.7245	0.7500	0.7167	0.7333
SC(λ=0.3)	0.6912	0.7500	0.6733	0.7117

	SVM with weighted vote			
Method	Accuracy	Sensitivity	Specificity	AUC
Raw data	0.6822	0.5833	0.7056	0.6444
KM	0.6284	0.6667	0.6122	0.6394
HC	0.6394	0.6283	0.6386	0.6334
SC(λ=0.1)	0.7091	**0.7833**	0.6844	**0.7339**
SC(λ=0.15)	**0.7559**	0.7000	**0.7678**	**0.7339**
SC(λ=0.2)	0.7176	0.7000	0.7178	0.7089
SC(λ=0.3)	0.6906	0.7500	0.6722	0.7111

Table 2.4: Classification performance on genes related to immune system

	SVM with majority vote			
Method	Accuracy	Sensitivity	Specificity	AUC
Raw data	0.6981	0.6167	**0.7156**	0.6661
KM	0.7079	0.7167	0.7067	0.7117
HC	0.6975	0.7500	0.6822	0.7161
SC(λ=0.1)	0.6911	0.7833	0.6622	0.7228
SC(λ=0.15)	0.7149	**0.8333**	0.6822	0.7578
SC(λ=0.2)	0.7065	0.7333	0.6933	0.7133
SC(λ=0.3)	**0.7399**	**0.8333**	0.7144	**0.7739**

	SVM with weighted vote			
Method	Accuracy	Sensitivity	Specificity	AUC
Raw data	0.6828	0.6167	0.6956	0.6561
KM	0.7079	0.7167	0.7067	0.7117
HC	0.6975	0.7500	0.6822	0.7161
SC(λ=0.1)	0.6911	0.7833	0.6622	0.7228
SC(λ=0.15)	0.7309	**0.8667**	0.6933	**0.7800**
SC(λ=0.2)	0.7225	0.7833	0.7033	0.7433
SC(λ=0.3)	**0.7476**	0.8333	**0.7244**	0.7789

Chapter 3

PRUNING DECISION TREE VIA MAX-HEAP PROJECTION

3.1 Introduction

Recently there has been a renewed interest in tree based methods among researchers in the field of data mining and machine learning Appel *et al.* (2013); Xu *et al.* (2014); Johnson and Zhang (2014); Bifet *et al.* (2010). This is probably due to the fact that the tree based methods have demonstrated impressive prediction performance in a great variety of recent applications like ranking Asadi and Lin (2013), recognition Mathias *et al.* (2014); Burgos-Artizzu *et al.* (2012), recommendation Amatriain (2013). On the other hand, unlike some nonlinear models such as kernel methods that generate predictions in a black-box fashion, the tree-based methods produce rules that can be easily interpreted and validated by domain knowledge or expert judgment, and can be flexibly revised based on new data or any other form of human knowledge intervention. The ease of interpretability usually leads to a higher likelihood of adaptability in real-world applications as critical decision-making tools. Furthermore, due to the unique hierarchical structure Lee *et al.* (2011), tree-based models support sequential prediction, which is more cost-effective since only a few variables are needed to produce a prediction, while in other predictive models such as regression models, all the variables are needed to make a prediction.

Despite the aforementioned advantages, the tree models have been suffering from some longstanding limitations. Overfitting is one of the well known notorious problems. For example, off-the-shelf decision tree learning algorithms such as C5.0 and Classification And Regression Trees (CART), (Breiman *et al.* (1984)) tend to create

over-complex trees that may not generalize to unseen data very well. These methods also have several hard-to-tune parameters which result in barriers for their usage in real-world applications. Furthermore, due to the lack of an explicit optimization formulation which could oversee the tree learning process and gear the learned tree toward the defined optimality (i.e., that ensures better generalizability), many of the existing methods rely on heuristics that heavily depend on the training data, leading to unstable or unreliable tree models.

To mitigate the overfitting problem, one school of thought is to employ an ad hoc pruning procedure to prune the tree structure in the hope to preserve the major skeleton that can generalize well on new data. Over the past decades, there have been a number of benchmark pruning methods being developed, such as reduced error pruning Quinlan (1987), pessimistic error pruning Quinlan (1986), etc. Many of these methods date back to 1980s and are based more or less on heuristics with statistical justification rather than an integrated optimization formulation Mingers (1989). This is a common limitation for other pruning methods such as the ones that attempt to exploit some information theoretic measure to prune a decision tree Mehta *et al.* (1995).

More recently the RuleFit Friedman and Popescu (2008) algorithm has been developed that extracts rules from trees since rules can provide powerful basis functions to approximate highly nonlinear functions. For instance, RuleFit aims to build a prediction model as a weighted combination of the nodes in the decision trees learned by an ensemble learning method, where each node in a tree is regarded as a rule function. This rule function takes the form as an indicator function indicating whether or not the conjunction of conditions associated with edges on the path from the root node of the tree to the node concerned is satisfied. By viewing that each decision tree is a collection of rule functions, RuleFit primarily focuses on how to select a

small subset of rules derived from multiple decision trees to best predict the response variable without giving consideration to the tree structure inherently existing among the rules. Without imposing the tree structure in learning the best rule combination, RuleFit leads to non-exclusive rules where other prediction mechanisms such as regression models need to be used to combine the learned rules to generate a prediction. Another work along this line of research is the Regularized Greedy Forest Johnson and Zhang (2014) which does take into account the structure of the trees. However, it does so by merely introducing into the objective function a regularization term which specifically concern with preventing the trees in the forest from growing too deep. Appel et al. Appel *et al.* (2013) made another attempt to mitigate the problem from another perspective. In their effort to speed up tree training, they prune those underachieving features through training on progressively larger subsets of samples. Therefore, there is still a lack of methodology, particularly, a lack of explicit optimization formulation, that can achieve optimal balance between the control of the tree complexity and the integrity of the tree structure, motivating the proposed research in this paper.

Recent developments in sparse learning and optimization enable us to address the tree pruning problem by formulating it as a sparse optimization problem. In this paper, we concern ourselves with post-pruning of a single decision tree. We follow the same practice adopted by the RuleFit by treating each node in the decision tree as a rule function, but take into consideration the tree structure that exists among the rules. Specifically, we propose a novel non-convex formulation for post-pruning of a decision tree that induces sparsity of weights of the nodes by appending a l_1 regularization term and requires the weights of nodes to satisfy the max-heap constraint. That is, as the tree structure implies, for each edge connecting a parent node and child node, the absolute value of the weight of the parent node should be

no less than the absolute value of the weight of the child node. It can be shown that the feasible set of this problem actually consists of a union of subspaces, leading to a non-convex optimization problem. In spite of the non-convex nature of this problem, we show that by the use of the concept known as proximal map, we can convert the non-convex optimization problem into a series of optimization problems that are not only convex but also smooth and can be efficiently solved by the method proposed in Liu *et al.* (2011). In this way, we could easily extend this sparse optimization model to study a broad class of general tree pruning problems by incorporating the prior information as a regularizer or constraint. Moreover, in order to overcome the model selection problem and enable robust performance of the proposed method, we propose a stability selection approach to select a robust weight vector among different subsampling and regularization parameters. We also prove that the selected weight vector will satisfy the tree constraint. Finally, through extensive experiments, we demonstrate that our proposed method achieves better predictive performance than many existing benchmark pruning methods across a wide range of real-world datasets.

The main contributions of this chapter are as follows: (1) We propose a novel non-convex formulation for post-pruning of a decision tree based on the ℓ_1 regularization and the max-heap constraint; (2) We develop an efficient algorithm to solve the proposed formulation based on the proximal method; (3) We propose a stability selection approach to improve the robustness of the resulting decision tree model; (4) We conduct extensive experiments using 19 data sets to evaluate the effectiveness of the proposed approach.

3.2 Background

In this section, we will review the basic concept about the rule function and show how it can be applied to represent a tree, which actually lays the first step to connect the tree models with sparse learning as adopted in RuleFit.

We first introduce our notation used throughout this chapter. We use boldface lower case letters (e.g., $\mathbf{d}, \mathbf{s}$) to denote vectors and use boldface upper case letters (e.g., $\mathbf{X}, \mathbf{Z}$) to denote matrices. Scalars and some other variables are denoted by lower or upper case letters (e.g., v, T).

3.2.1 Rule Function

Let $T = (V, E)$ be a decision tree obtained through some decision tree training procedure such as CART. It consists of a node set $V = \{v_0, v_1, v_2, \cdots, v_p\}$ and an edge set $E = \{(v_i, v_j) | v_j \text{ is a child node of } v_i\}$. Each node v_i represents a "rule function" defined by the conjunction of all conditions associated with the edges on the path from the root of the tree to that node. As illustrated in Figure 1, the rule functions at nodes 1, 4, 10 are

$$
\begin{aligned}
v_1(\mathbf{d}) &= I\left(\mathbf{d}[20] \leq 0.5\right); \\
v_4(\mathbf{d}) &= I\left(\mathbf{d}[20] \leq 0.5\right) \cdot I\left(\mathbf{d}[5] > 1\right); \\
v_{10}(\mathbf{d}) &= I\left(\mathbf{d}[20] > 0.5\right) \cdot I\left(\mathbf{d}[6] > 0\right) \cdot I\left(\mathbf{d}[20] > 1.5\right).
\end{aligned}
$$

where $\mathbf{d}$ is a row vector representing a sample and I is the indicator function. For simplicity, we only show the cases where the nodes are associated with continuous features. The rule function can be defined with respect to categorical features, e.g., instead of specifying ranges for continuous features, non-trivial subset of domain of the corresponding categorical features can be used in defining the rule functions.

With the use of the rule function, a tree can be represented by a collection of rule

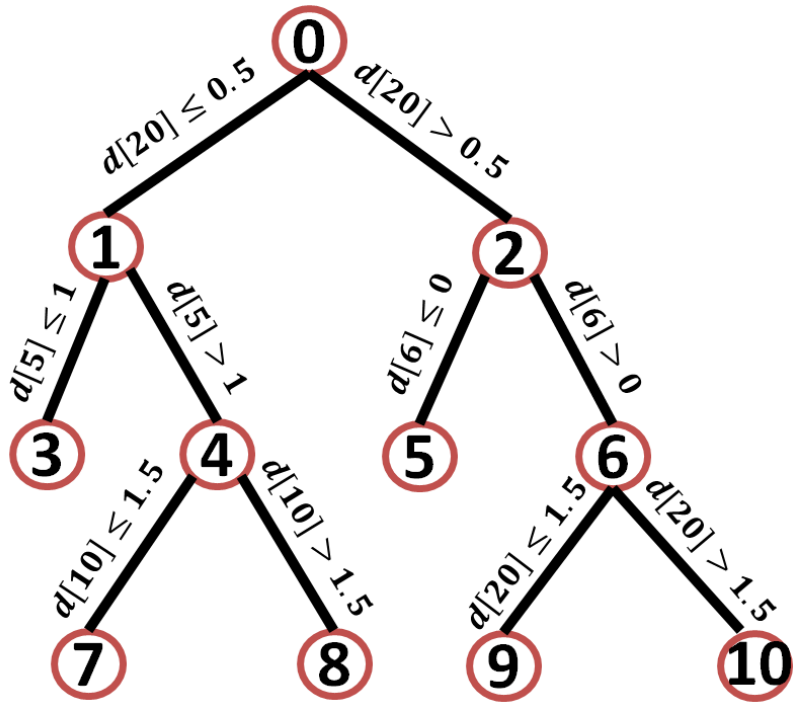


Figure 3.1: A decision tree. The rule function corresponding to each node can be represented as the product of indicator functions associated with edges on the path connecting the node to the root.

functions (each node v_i contributes a rule function except the root node). This collection of rule functions can be viewed as a new basis so any sample can be represented using this basis. Specifically, suppose that we have a dataset $\mathbf{D} = [\mathbf{d}_1; \mathbf{d}_2; \cdots; \mathbf{d}_n]$ with their response denoted as $\mathbf{y} = [y_1; y_2; \cdots; y_n]$ where n is the number of samples in the dataset (here we use the MATLAB syntax ";" to denote vertical concatenation). Then, denote $\mathbf{x}_i = v_i(\mathbf{D})$ which is actually the projection of the dataset onto the basis v_i. v_i maps each row of $\mathbf{D}$ into 0 or 1 depending on whether the corresponding sample satisfies all the conditions included in v_i. Thus, with $p + 1$ nodes in the tree T and their corresponding the rule functions, we can get a new representation of the original dataset $\mathbf{D}$ which we write as $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_p]$ while the response is still $\mathbf{y}$ (the MATLAB syntax "," is used to denote horizontal concatenation). By using this new

representation of the original dataset $\mathbf{D}$, we can formulate rule selection problem as a feature selection problem as adopted in RuleFit. Note that since there is no rule associated with the root, we define $v_0(\mathbf{d}) = 1$ and x_0 as a column vector of 1s.

3.2.2 RuleFit

RuleFit seeks to learn an ensemble of rule functions derived from a set of trees learned by an ensemble learning method such as random forest. It further uses linear combination of the rules to generate predictions, so each rule function v_i is associated with a weight w_i whose magnitude reflects the significance of the corresponding rule. With a collection of K rules $\{v_1, v_2, \cdots, v_K\}$ derived from a set of M trees, $\{T_1, T_2, \cdots, T_M\}$, we can formulate RuleFit as the following optimization problem:

$$\min_{\{w\}_{k=1}^{K}, b} \quad \sum_{i=1}^{n} L\left(y_i, b + \sum_{k=1}^{K} w_k v_k(\mathbf{d}_i)\right) + \lambda \sum_{k=1}^{K} |w_k| \tag{3.1}$$

where L is a general loss function. As the formulation of RuleFit implies, in its course of seeking a small subset of rules to explain the observed response, it completely ignores the inherent tree structure existing among the rules. Using the package provided by its authors, we thoroughly studied the RuleFit on some real-world datasets and found that the best prediction performance is usually attained when the collection of trees mainly consists of tree stumps and the selected set of rules is large. This is probably due to the fact that the decision tree tends to overfit the training samples when it is built to a great depth. The l_1-norm regularization term used in RuleFit does not effectively control the structural risk since nodes deep down the tree tend to be selected in order to have a sparse solution with small loss instead of those nodes that are closer to the root which are less complex and are less prone to overfitting. Furthermore, without imposing the tree structure in learning the best rule combination, RuleFit leads to non-exclusive rules where other prediction mechanisms

such as regression models need to be used to combine the learned rules to generate a prediction.

3.3 The Proposed Decision Tree Pruning Algorithm

In this paper, we focus on post-pruning a single decision tree. The unique aspect of our idea of pruning a decision tree is to parameterize the pruning process by using the rule functions to represent a tree and further translating the hierarchical structure of the tree into max-heap constraints. By parameterizing the tree learning process, it paves the way for developing sparse learning formulations. Specifically, recall that each node (except the root node) corresponds to a rule while each rule is associated with a weight, the max-heap constraint requires that the magnitude of the weight of an ancestor node to be greater than or equal to that of an descendant node. In other words, for any i, j $(1 \leq i, j \leq p)$, if (v_i, v_j) is an edge in T, then the weight vector $\mathbf{w} = (w_0, w_1, \cdots, w_p)^T$ satisfies max-heap constraint if $|w_i| \geq |w_j|$. For notational convenience, let us denote the set of weights that satisfy the max-heap constraint as $P = \{\mathbf{w} \mid |w_i| \geq |w_j|, \text{if } (v_i, v_j) \in E\}$. Pruning the tree could be achieved by imposing both the sparsity constraint and max-heap constraint on $\mathbf{w}$ since once the weight of a node becomes zero, the weight of all its descendants node will be zero.

3.3.1 Proposed Formulation

Following the aforementioned idea, we can formulate the problem of tree pruning via max-heap projection as the optimization problem below:

$$\min_{\mathbf{w},b} L(\mathbf{y}, F(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_p; \mathbf{w}, b)) + \lambda \|\mathbf{w}\|_1 \quad s.t. \mathbf{w} \in P, \tag{3.2}$$

where λ is the regularization parameter controlling the sparsity of $\mathbf{w}$, L is a general loss function with respect to x_i, $\mathbf{w}$ and b, and prediction function F takes the form:

$$F(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_p; \mathbf{w}, b) = b\mathbf{1}_n + \sum_{i=0}^{p} w_i x_i.$$

Once a decision tree has been learned, the features and the threshold associated with each node is fixed. In other words, all $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_p$ are fixed during the decision tree post-pruning stage. For simplicity of notation, we denote $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_p]$ and write $L(\mathbf{y}, F(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_p; \mathbf{w}, b))$ as $H_{(\mathbf{X},\mathbf{y})}(\mathbf{w}, b)$. With this, we can rewrite (3.2) as

$$\min_{\mathbf{w},b} H_{(\mathbf{X},\mathbf{y})}(\mathbf{w}, b) + \lambda \|\mathbf{w}\|_1$$
$$s.t. \mathbf{w} \in P. \tag{3.3}$$

Note that to preserve integrity of the tree structure, we keep the constant x_0 and its coefficient w_0 in our model. However, their effects can be compensated by the intercept term on which no constraint has been imposed. In our experiments, we use least square loss for $H_{(\mathbf{X},\mathbf{y})}(\mathbf{w}, b)$.

3.3.2 Proposed Algorithm

Due to the fact that the feasible set of the resulting optimization problem formulated above can be represented as a union of multiple non-overlapping convex cones, it is essentially a non-convex problem. One effective approach to solving this problem is to employ the General Iterative Shrinkage Threshold (GIST) Gong *et al.* (2013) framework. At the heart of GIST framework lies in solving the proximal map for the problem through which a sequence of points will be generated towards a local optimal solution under certain conditions. We found that by decomposing the proximal map into an element-wise product of signs and magnitudes, finding proximal map can be converted into a smooth convex problem which can be efficiently solved by some existing tools Liu *et al.* (2009).

Let

$$\sigma(\mathbf{w}) = \begin{cases} \lambda\|\mathbf{w}\|_1, & \text{if } \mathbf{w} \in P \\ +\infty, & \text{if } \mathbf{w} \notin P. \end{cases} \tag{3.4}$$

At the point $\left(\mathbf{w}^k, b^k\right)$, the proximal map is

$$\begin{aligned} &\left(\mathbf{w}^{k+1}, b^{k+1}\right) \\ &= \arg\min_{\mathbf{w},b} \left\langle \nabla_{\mathbf{w}} H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right), \mathbf{w} - \mathbf{w}^k \right\rangle \\ &+ \left\langle \nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right), b - b^k \right\rangle + \frac{t^k}{2}\|\mathbf{w} - \mathbf{w}^k\|_2^2 \\ &+ \frac{t^k}{2}\|b - b^k\|_2^2 + \sigma(\mathbf{w}) \end{aligned} \tag{3.5}$$

where $t^k > 0$ can be found through line search. After rearranging the terms and removing some constant terms, we have

$$\begin{aligned} &\left(\mathbf{w}^{k+1}, b^{k+1}\right) \\ &= \arg\min_{\mathbf{w},b} \frac{t^k}{2} \left\| \mathbf{w} - \left(\mathbf{w}^k - \frac{\nabla_{\mathbf{w}} H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right)}{t_k} \right) \right\|_2^2 + \sigma(\mathbf{w}) \\ &+ \frac{t^k}{2} \left\| b - \left(b^k - \frac{\nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right)}{t^k} \right) \right\|_2^2 \end{aligned} \tag{3.6}$$

Obviously, $b^{k+1} = b^k - \frac{\nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right)}{t^k}$. Let $\mathbf{u}^k = \mathbf{w}^k - \frac{\nabla_{\mathbf{w}} H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right)}{t_k}$. Finding $\mathbf{w}^{k+1}$ amounts to solving the following problem:

$$\begin{aligned} \mathbf{w}^{k+1} = &\arg\min_{\mathbf{w}} \frac{1}{2}\left\|\mathbf{w} - \mathbf{u}^k\right\|_2^2 + \frac{\lambda}{t^k}\|\mathbf{w}\|_1 \\ &s.t. \quad \mathbf{w} \in P \end{aligned} \tag{3.7}$$

Next, we show that the optimal solution $\mathbf{w}^{k+1}$ yielded by (3.7) shares the same sign as $\mathbf{u}^k$ in an element-wise way. To simplify notation, let us define sign$(\cdot)$ as the sign function of a vector that returns the element-wise sign of the vector.

Lemma 1. *Let* $\mathbf{w}^{k+1}$ *be the optimal solution yielded by (3.7), then we have* $\text{sign}\left(\mathbf{w^{k+1}}\right) = \text{sign}\left(\mathbf{u^k}\right)$.

Proof. This can be easily verified by contradiction. Suppose that the j-th element of the $\mathbf{w}^{k+1}$, w_j^{k+1} does not have the same sign as the j-th element of $\mathbf{u}^k$, u_j^k. Let $\hat{\mathbf{w}}^{k+1}$ be a column vector with the i-th element denoted as $\hat{w}_i^{k+1}$. Let $\hat{w}_i^k = \hat{w}_i^{k+1}, \forall i \neq j$, $\hat{w}_j^{k+1} = -\hat{w}_j^{k+1}$. Since the magnitude of each element of $\hat{\mathbf{w}}^{k+1}$ is the same as that of corresponding entry in $\mathbf{w}^{k+1}$, $\hat{\mathbf{w}}^{k+1}$ is a feasible solution and has the same 1-norm as $\mathbf{w}^{k+1}$, but $\left\|\mathbf{w}^{k+1} - \mathbf{u}^k\right\|_2^2 \geq \left\|\hat{\mathbf{w}}^{k+1} - \mathbf{u}^k\right\|_2^2$. Thus $\mathbf{w}^{k+1}$ is a less optimal solution than $\hat{\mathbf{w}}^{k+1}$, which is in contradiction with our assumption. □

Denote the vector obtained from taking element-wise absolute value of $\mathbf{u}^k$ as $\mathbf{u}^{k^+}$, we have $\mathbf{u}^k = \text{sign}(\mathbf{u}^k) \circ \mathbf{u}^{k^+}$, where "$\circ$" represents element-wise multiplication. Knowing that the optimal solution $\mathbf{w}^{k+1}$ shares the same sign as $\mathbf{u}^k$, we can further convert the problem (3.7) to

$$\begin{aligned} \mathbf{w^{k+1}}^+ = & \arg\min_{\mathbf{w}} \frac{1}{2}\left\|\mathbf{w} - \mathbf{u}^k\right\|_2^2 + \frac{\lambda}{t^k}\mathbf{w}^T\mathbf{1}_{p+1} \\ s.t. \quad & \mathbf{w} \succeq 0 \\ & w_i \geq w_j, \quad \forall (v_i, v_j) \in E, 0 \leq i, j \leq p. \end{aligned} \tag{3.8}$$

The optimal solution $\mathbf{w}^{k+1} = \text{sign}(\mathbf{u}^k) \circ \mathbf{w^{k+1}}^+$. The problem (3.8) can be efficiently solved by the method proposed in Liu *et al.* (2011). The framework of the algorithm is described in Algorithm 3.

As described in Liu *et al.* (2011), the key step in solving the problem of finding proximal map is to recursively identify the maximal root-tree and remove it from the original tree. It can be shown that such a process can lead to unique optimal solution. As mentioned in the paper, this process has an expected linear complexity. We refer interested readers to Liu *et al.* (2011) for more details.

Algorithm 3 Max-heap Based Decision tree Pruning Algorithm

Initialization:

$X = \{x_0, x_1, \ldots, x_p\}$, $\lambda \in \mathbb{R}$, $\eta > 1$,initial weight and intercept $\mathbf{w}^{(0)}$, $b^{(0)}$, t_{min}, t_{max} with $0 < t_{min} < t_{max}$.

$k \leftarrow 0$

repeat

  $t^{(k)} \in [t_{min}, t_{max}]$

  repeat

    $\mathbf{u}^k = \mathbf{w}^{(k)} - \frac{\nabla_{\mathbf{w}} H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^{(k)}, b^{(k)}\right)}{t^{(k)}}$

    $\mathbf{u}^{(k)^+} = \text{abs}\left(\mathbf{u}^{(k)}\right)$

    $\mathbf{w}^{(k+1)^+} \leftarrow$ solution to the problem (3.7)

    $\mathbf{w}^{(k+1)} \leftarrow \text{sign}(\mathbf{u}^{(k)}) \circ \mathbf{w}^{(k+1)^+}$

    $\mathbf{b}^{(k+1)} = b^{(k)} - \frac{\nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^{(k)}, b^{(k)}\right)}{t^{(k)}}$

    $t^{(k)} \leftarrow \eta t^{(k)}$

  until some line search criterion is satisfied.

  $k \leftarrow k + 1$

until some stop criterion is satisfied

Output: $\mathbf{w}^{(k)}, b^{(k)}$.

3.4 Stability Selection

In order to optimize the decision tree structure, we have proposed a max-heap projection method to solve the sparse optimization problem with the tree constraint in the previous section. The sparsity of the weight vector, which determines the tree structure, is controlled by the regularization parameter. In practice, the selection of the regularization parameter is usually data-dependent and sometimes sensitive to the noise of the data. To obtain a more robust tree model, we propose to apply stability

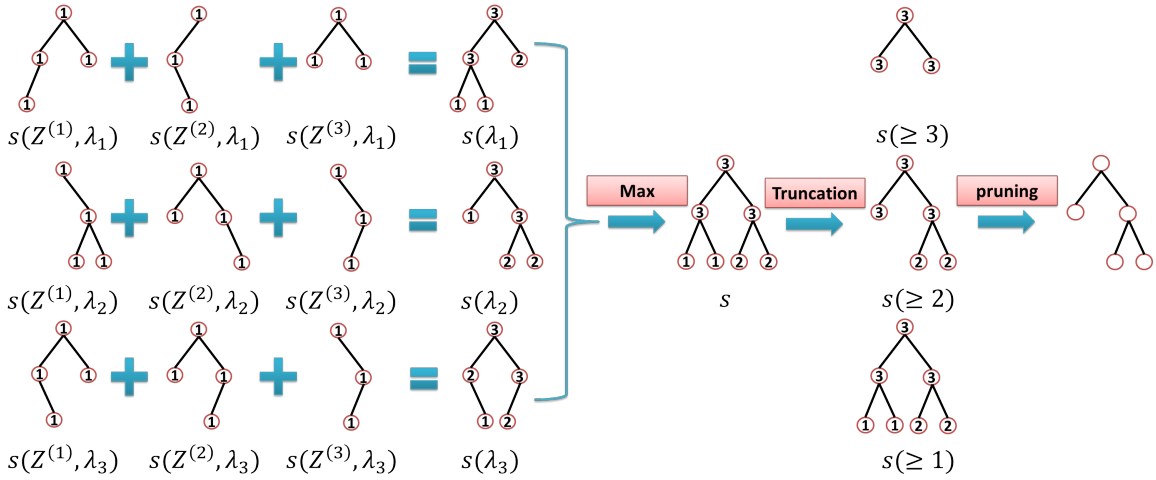


Figure 3.2: Stability selection. Given subsampled data sets $\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, \mathbf{Z}^{(3)}$ and regularization parameters $\{\lambda_1, \lambda_2, \lambda_3\}$, we first calculate $\{\mathbf{s}(\lambda_1), \mathbf{s}(\lambda_2), \mathbf{s}(\lambda_3)\}$ by adding from $\mathbf{s}(\mathbf{Z}^{(1)}, \lambda_i)$ to $\mathbf{s}(\mathbf{Z}^{(3)}, \lambda_i)$. Then we apply the max operator on $\{\mathbf{s}(\lambda_1), \mathbf{s}(\lambda_2), \mathbf{s}(\lambda_3)\}$ to obtain $\mathbf{s}$. By truncating different values on $\mathbf{s}$, we get several candidates of the tree structure. The tree structure will be selected by evaluating the performance on the pruning data set.

selection for model selection. Stability selection is a technique for feature selection. But here, since each feature represents a tree node, we adapt it to be a method for tree structure selection that guarantees the selected nodes constitute a tree.

Specifically, given a data set $\mathbf{X}$ and a response variable vector $\mathbf{y}$, we split $\mathbf{Z} = \{\mathbf{X}, \mathbf{y}\}$ to three parts, training data set $\mathbf{Z}_{tr}$, pruning data set $\mathbf{Z}_{pr}$ and testing data set $\mathbf{Z}_{te}$. Since the tree pruning process is equivalent to the learning of the weight vector $\mathbf{w}$, to account for the uncertainty of $\mathbf{w}$ induced by noise in training data set $\mathbf{Z}_{tr}$ and different choices of λ, the basic idea of stability selection is to resample the training data set $\mathbf{Z}_{tr}$ many times and learn the weight vector $\mathbf{w}$ on these resampled datasets across a range of λ. Then, the uncertainty of the elements in $\mathbf{w}$ can be evaluated

(i.e., via the support vector of $\mathbf{w}$ that will be defined below) and only the elements that tend to be selected most frequently should be kept in the final tree structure.

3.4.1 Select the Tree Structure for a Fixed λ

Given any subsampled training set $\mathbf{Z}^{(i)} = \{\mathbf{X}^{(i)}, \mathbf{y}^{(i)}\}$ from $\mathbf{Z}_{tr}$, the weight vector can be obtained as follows:

$$\mathbf{w}(\mathbf{Z}^{(i)}, \lambda) = \underset{\mathbf{w} \in P}{\operatorname{argmin}} H_{\mathbf{Z}^{(i)}}(\mathbf{w}, b) + \lambda \|\mathbf{w}\|_1$$

We assume there are s subsampled training data sets, denoted by $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(s)} \subset \mathbf{Z}_{\text{tr}}$. The number of data points in each $\mathbf{Z}^{(i)}$ is about half of the number of data points $\mathbf{Z}_{tr}$. Given a fixed λ, for each $\mathbf{Z}^{(t)}$ there is a weight vector $\mathbf{w}(\lambda, \mathbf{Z}^{(t)})$. The support of the weight vector, which means the locations of nonzero entries in the weight vector, determines the tree structure. Let $\mathbf{s}(\lambda, \mathbf{Z}^{(t)})$ denote a binary vector such that $\mathbf{s}(\lambda, \mathbf{Z}^{(t)}) = 1$ if $\mathbf{w}(\lambda, \mathbf{Z}^{(t)}) \neq 0$, $\mathbf{s}(\lambda, \mathbf{Z}^{(t)}) = 0$ otherwise. Examples of the support vectors are shown in Figure 3.2. Clearly $\mathbf{s}(\lambda, \mathbf{Z}^{(t)})$ belongs to P and $\mathbf{s}(\lambda, \mathbf{Z}^{(t)})$ induces the same tree structure as $\mathbf{w}(\lambda, \mathbf{Z}^{(t)})$ does.

Given a fixed λ, we aim to learn a support vector $\mathbf{s}(\lambda)$ from $\{\mathbf{s}(\lambda, \mathbf{Z}^{(1)}), \ldots, \mathbf{s}(\lambda, \mathbf{Z}^{(s)})\}$. Since the frequency of the components of the weight vector reflects its importance and probability, we simply define the new support vector as a summation of these support vectors,

$$\mathbf{s}(\lambda) := \sum_{t=1}^{s} \mathbf{s}(\lambda, \mathbf{Z}^{(t)}). \tag{3.9}$$

We can show that the new support vector $\mathbf{s}(\lambda)$ still has a tree structure.

Proposition 3.4.1. *The new support vector* $\mathbf{s}(\lambda) \in P$.

Proof. For any t, $1 \leq t \leq s$, and $(i, j) \in E$ we have $\mathbf{s}_i(\lambda, \mathbf{Z}^{(t)}) \geq \mathbf{s}_j(\lambda, \mathbf{Z}^{(t)})$. Adding from $t = 1$ to $t = s$, we have $\mathbf{s}_i(\lambda) \geq \mathbf{s}_j(\lambda)$ for any $(i, j) \in E$, which indicates that $\mathbf{s}(\lambda) \in P$. □

3.4.2 Select the Tree Structure for Multiple λ's

Next we would like to select a robust tree structure among multiple support vectors learned from different λ. Given a set of regularization parameters $\{\lambda_1, \ldots, \lambda_k\}$, there are k support vectors $\mathbf{s}(\lambda_1), \ldots, \mathbf{s}(\lambda_k)$, which can be computed by (3.9). In this case, the max selection rule is widely used and has a strong theoretical guarantee Meinshausen and Bühlmann (2010). Therefore, we propose to select the maximum value among $\{\mathbf{s}(\lambda_1), \ldots, \mathbf{s}(\lambda_k\}$ for each node of the tree. Specifically, we define the new support vector as follows:

$$\mathbf{s}_i := \max\{\mathbf{s}_i(\lambda_1), \ldots, \mathbf{s}_i(\lambda_k)\}, \ \forall i.$$

Interestingly, the new support vector still has a tree structure.

Proposition 3.4.2. *Let* $\mathbf{s}$ *be formed by taking maximum at each entry over the support vectors computed from multiple* λ*'s. Then* $\mathbf{s} \in P$.

Proof. It suffices to prove that for any edge $(i, j) \in E$, we have $\mathbf{s}_i \geq \mathbf{s}_j$. Without loss of generality, we assume $\mathbf{s}_i = \mathbf{s}_i(\lambda_{l_i})$ and $\mathbf{s}_j = \mathbf{s}_j(\lambda_{l_j})$ where $1 \leq l_i, l_j \leq p + 1$. By the construction of $\mathbf{s}_i$, we have $\mathbf{s}_i = \mathbf{s}_i(\lambda_{l_i}) \geq \mathbf{s}_i(\lambda_{l_j})$. Since $\mathbf{s}(\lambda_{l_j}) \in P$, we have $\mathbf{s}_i(\lambda_{l_j}) \geq \mathbf{s}_j(\lambda_{l_j})$. The conclusion follows by combining these two inequalities. □

By truncating different values on the support vector, we obtain several candidates of the tree structure. The final tree structure will be selected by evaluating the performance on the pruning data set. The whole procedure is illustrated in Figure 3.2.

3.5 Related Work

Overfitting is a major problem with decision tree methods Safavian and Landgrebe (1991). This is due to the fact that, on the one hand, decision tree methods

indeed provide powerful capability for capturing nonlinear interactions among variables that sufficiently represent the training data, but on the other hand, this good sensitivity to nonlinear patterns embedded in data inevitably brings in instability since small perturbation of the training data could result in completely different tree structures. Existing methods for alleviating the overfitting problem of tree learning can be roughly divided into two areas, one is the ensemble learning and the onther one is tree pruning Rastogi and Shim (1998). The ensemble learning grows a collection of trees and generates prediction by averaging these trees. Methods that fall into this category include Random Forest Breiman (2001), Adaboost, Gradient Boost Decision Tree Friedman (2001), Regularized Greedy Forest Johnson and Zhang (2014). The ensemble learning grows a collection of trees and generates prediction by averaging these trees. Methods that fall into this category include Random Forest Breiman (2001), Adaboost Freund *et al.* (1999) , Gradient Boost Decision Tree Friedman (2001), Regularized Greedy Forest Johnson and Zhang (2014), etc. These ensemble methods differ from one another in several aspects such as how to select the feature at each tree node, which samples to use to learn a new tree, what objective functions to be optimized for generation of a new tree, and what strategies should be employed for obtaining the linear combination parameter for each of the ensemble member.

While ensemble learning methods aim to robustify the tree based methods by averaging, it loses the trait of interpretability. Pruning is another major mechanism that has been widely adopted to prevent decision trees from overfitting. There are basically two broad categories of pruning algorithms, pre-pruning and post-pruning. Pre-pruning strategies keep a decision tree from growing further when some conditions are no longer satisfied. Algorithms that fall in this class include ID3 Quinlan (1986) which chooses a feature that maximizes information gain to split on and employs chi-

squared test as the criterion to decide when to stop growing a new branch. CHAID Kass (1980) is another algorithm that employed the strategy of pre-pruning.

Post-pruning decision tree, however, is viewed as a more favorable approach. Some of the popular methods include: Lee *et al.* (2008) which chooses a feature that maximizes information gain to split on and employs chi-squared test as the criterion to decide when to stop growing a new branch. CHAID Kass (1980) is another algorithm that employed the strategy of pre-pruning.

- Reduced error pruning (REP) Quinlan (1987). Start with the complete tree and run the samples from the pruning set through it. For each internal node, it compares the number of classification errors made by the node when it is treated as a leaf node, with the number of classification errors when the subtree rooted at this node is kept. Prune the sub-tree if the number of errors obtained from treating it as a sub-tree is greater than the number of errors obtained from treating it as a leaf node.

- Pessimistic error pruning (PEP) Quinlan (1986). It uses only the samples from the training set to prune the tree. It penalizes the number of classification errors of an internal node by adding 0.5 to it when it is treated as a leaf node and by adding half of the leaf nodes in the sub-tree when it is treated as a root of a subtree. The subtree rooted at the node will be kept if the penalized error computed from it being treated as a leaf node is greater than the penalized error obtained from it being treated as sub-tree plus the standard error of misclassifications of the subtree.

- Cost complexity pruning (CCP)Breiman *et al.* (1984). It starts with the complete tree. Based on the training set, it calculates the ratio between the number

of classification errors that would be reduced by keeping the sub-tree and number of leaf nodes that would be increased when keeping the sub-tree. It selects the internal node with the smallest ratio to prune and runs the same process on the resulting new tree. A series of trees with decreasing size will be generated by repeating this process until there is no more sub-tree to prune. The true error rate for each tree is estimated and the tree with smallest error rate is chosen as the final pruned tree. In Breiman *et al.* (1984), the authors propose two ways of estimating the true error rate, one based on cross-validation sets, the other on an independent pruning set.

- Minimum error pruning (MEP) Niblett and Bratko (1987). For each node v_i, it calculates the expected error rate which is defined on the independent pruning set as

$$E_i = \frac{e_i + c - 1}{\mathbf{1}_n^T \cdot \mathbf{x}_i + c},$$

where e_i is the number of classification errors at this node and c is the total number of classes. The decision of whether the sub-tree rooted at the node should be pruned is made based upon whether the expected error rate at the node is greater than the weighted sum of expected error rates of its two child nodes with the weight determined by sample number.

3.6 Experiments

In this section, we evaluate the proposed method for pruning decision trees on a variety of real-world datasets of different sizes and levels of difficulty from UCI machine learning repository Lichman (2013) and some other sources. We compare the predictive performance of our proposed method with that of the baseline methods on both task of classification and regression. For classification, we focused on binary

classification. Multi-class classification problems can be simply converted to multiple binary ones by conducting pair-wise binary classification or one versus rest binary classification.

3.6.1 Datasets and Experimental Setup

In our experiments, all the datasets that are used in the task of classification are from the UCI machine learning repository. All missing values were imputed with column mean. In the training stage, we use the scikit-learn package Pedregosa *et al.* (2011) to build the decision tree. It implements an optimized version of CART. For classification, samples from each of the two classes are equally divided into three folds. One is used for building the basic decision tree, one for pruning and one for testing. For a fair evaluation of different pruning methods, we carry out this random split ten times and report the average performance over these ten random splits. For our method, we learn the weights of the nodes in decision trees on the training set. Table 3.1 shows the data sets, their statistics, respective task types and the average number of nodes in the decision trees built on training sets.

For regression, we simply divide all the samples equally into three folds, one for training the initial decision tree, one used as the independent pruning set, and the last fold used for testing. This random split is also carried out for ten times and the average performance is reported. In building the initial decision tree for pruning, we set the maximum depth of the tree to 15. We use the squared root of mean squared error (RMSE) as the metric for evaluation of performance of different pruning methods for regression. To reduce the effect of the scale of response, we center the response vector to zero and then divide it by its infinity norm such that the range of the response for all the samples is $[-1, 1]$.

In both classification and regression, a series of values of λ has been tried and the

model selection is performed based on the independent pruning set. In our experiments, λ is typically set to be $r\lambda_{max}$ where λ_{max} is the minimum value of λ at which the corresponding lasso problem attains a all zero solution and $r = 2^{[-15:0.25:-6]}$.

In making predictions, we use weighted sum of rule functions of the tree nodes as the discriminant function. Note that this discriminant function can be written into an equivalent form of weighted sum of rule functions corresponding only to the leaf nodes. As is pointed out in Johnson and Zhang (2014), the rule function of an internal node can be decomposed into the sum of the rule functions of its two child nodes, i.e. $v_i(x) = v_j(x) + v_k(x)$ where node i is an internal node and j and k are its two child nodes. By applying this rule recursively over all the internal nodes of the pruned tree, the weights associated with all the internal nodes will eventually pass down onto leaf nodes. When it comes to making a prediction for a specific test sample, only one leaf node will actively participate in determining its label since the values of rule functions of all other leaf nodes at this sample are zero.

3.6.2 Baseline Methods

For comparison in terms of prediction performance of post-pruned decision trees, we use several well-known methods including Reduced Error Pruning (REP) Quinlan (1987), Pessimistic Error Pruning (PEP) Quinlan (1986), Cost Complexity Pruning (CCP)Breiman *et al.* (1984), and Minimum Error Pruning (MEP) as the baseline methods. We refer interested readers to Esposito *et al.* (1997); Mingers (1989) for a comprehensive analysis of these methods.

In the task of regression, we replace the classification error in REP, PEP and CCP by squared difference between the predicted response and the ground-truth response. As for MEP, we omit it from our experiments on regression since the definition of expected error rate explicitly involves the number of classes.

3.6.3 *Results*

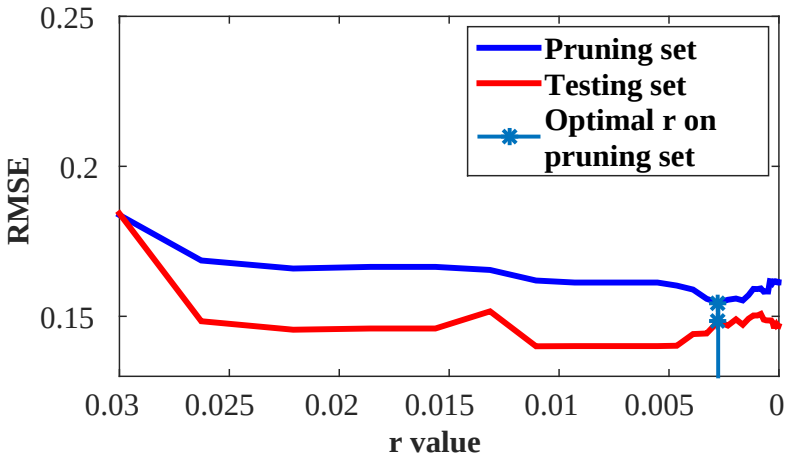


Figure 3.3: Error curves when $r = \lambda/\lambda_{\max}$ varies

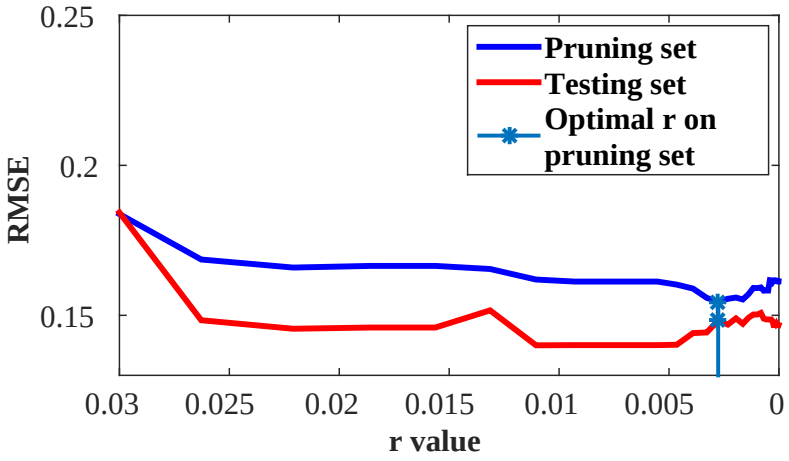


Figure 3.4: Error curves of stability selection

Table 3.1: Statistics of datasets used in the task of classification and performance of various methods. The classification performance is measured in terms of the Area Under Curve (AUC)

datasets	# features	# samples	REP(AUC)	PEP(AUC)	CCP(AUC)	MEP(AUC)	Maxheap(AUC)
sonar	60	208	0.7031	0.7167	0.7021	0.7124	**0.7233**
biodeg	41	1055	0.5212	0.7948	0.7616	0.7842	**0.8034**
bank note	4	1372	0.9109	0.9764	0.9769	0.9784	**0.9796**
Musk1	166	476	0.6663	0.7486	0.7378	0.7374	**0.7576**
Musk2	200	200	0.5	0.9389	0.9266	0.9364	**0.9468**
EEGEyeState	14	14980	0.6628	0.8491	0.8445	0.8488	**0.8513**
LSVT	313	126	0.6703	0.7071	0.7037	0.7050	**0.7154**
masses	5	961	0.8171	0.8390	0.8431	0.8699	**0.8843**
arcene	10000	200	0.6386	0.6484	0.6404	**0.6517**	0.6502
madelon	500	2600	0.7136	0.7693	0.767	0.7725	**0.7762**

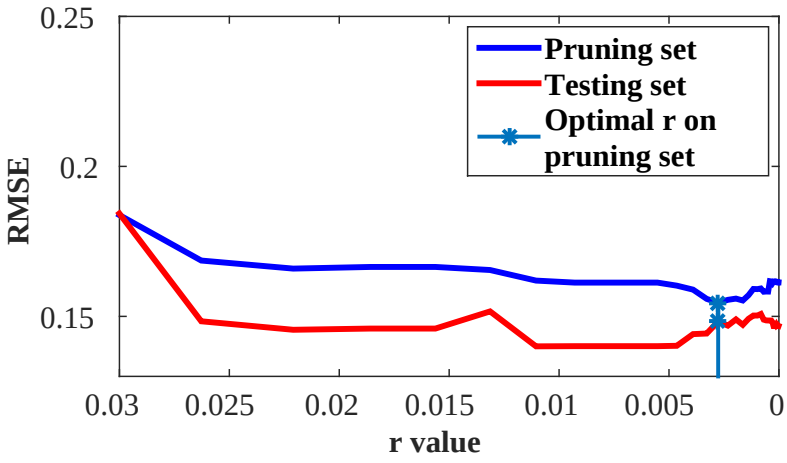


Figure 3.4: Error curves when $r = \lambda/\lambda_{\max}$ varies

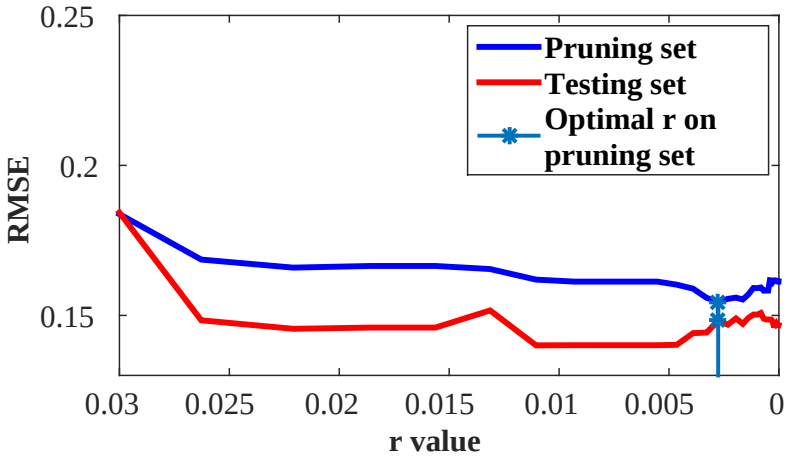


Figure 3.5: Error curves of stability selection

Figure 3.6: (a) and (c) show the Root Mean Square Error (RMSE) curve when $r = \lambda/\lambda_{\max}$ varies on two data sets respectively. (b) and (d) show the RMSE error curves of stability selection when the number of selected features varies on two data sets respectively. (a) and (c) are generated using one random split of the dataset "housing". (c) and (d) are generated using another random split of the same dataset. It can be seen that 1) the performance of stability selection is quite stable; (2) the models selected by stability selection perform well on the testing set, which is comparable or even better than the performance of the models selected by cross validation.

The prediction performance in the task of classification is shown in Table 3.1. We can see from the table the prediction performance of our proposed method is better than the best performance of the baseline methods on most of the datasets. The average number of nodes in the decision tree before and after pruning by different methods are shown in Table 3 in the supplement. We can also see that REP generally performs worse than other methods and usually results in an over-pruned decision tree, which is consistent with the observation made in Esposito *et al.* (1997).

The prediction performance in the task of regression is shown in Table 3.2. We can see from the table that our proposed method also achieves the best performance on most of the datasets. In most cases, unlike the classification, REP did not yield an over-pruned tree in task of regression. Our method generally produce smaller decision trees with better prediction performance. From Table 3 in the supplement, it is not difficult to observe that PEP prunes the entire tree in most cases. This is not surprising since in this method, the decision regarding whether a sub-tree should be pruned or not depends on a standard error term, while the magnitude of this standard error term relies on the scale of the response, making this method inappropriate for the task of regression.

We think the better performance obtained by our method can be attributable to the following facts. First, although all the methods start with the same tree to prune, the traditional methods base their effort to optimize certain heuristic metric on a greedy approach. The effect of pruning a subtree is irreversible. Once a subtree is pruned, it can no longer become part of the real optimal solution. Even for CCP which selects the best subtree from a set of sequentially pruned decision trees, chances are that a subtree that belongs to the real optimal solution has been pruned prematurely. By estimating the optimally pruned tree on the basis of solving an optimization problem directly related to predictive performance, our method is able to avoid such

an issue. Second, in calculating AUC, all the traditional methods weight the leaf nodes by the proportion of samples from the majority class among all the samples that fall into this node. In contrast, the weight of each node is determined by the solution to the formulated optimization problem in our approach.

3.6.4 Stability Selection

In our experiments, we found that the model that gives the best prediction performance on the independent pruning set does not usually yield the best performance on the testing set, even though the pruning set consists of 1/3 of samples in every dataset. This problem becomes even more evident when the number of samples in the dataset is small. This motivates us to look for a method that produces a more stable performance on the testing set when using an independent pruning set to determine the structure of the tree. We proposed to use stability selection to solve this problem of model selection. Furthermore, we have proved that the model returned by this stability selection procedure still preserves the tree structure.

To demonstrate the effectiveness of stability selection, we use the dataset "housing" as an example on which our method did not achieve the best performance. Figure 3.6 shows that how RMSE changes on the pruning set and testing set, respectively, by using the different models produced by different λ values in our experiments and models given by stability selection based on two different random splits of data. We can see from this figure that the best model selected by the independent pruning set using cross validation does not perform well on the testing set. In contrast, the best model selected by the independent pruning set using stability selection is very close to the best model on the testing set. Furthermore, it is very interesting to observe that the best model given by cross validation is relatively "dense" and is located at a place where the pruning set rMSE and testing set rMSE are in the process of diverging and

converging. As a contrast, the best model given by the pruning set based on stability selection is more "sparse" and also very close to the best model on the testing set. These observations imply that the stability selection can potentially yield the smallest tree while still achieving the best prediction performance.

3.6.5 Computation Time

It appears that our proposed method is the slowest one among all the methods since our method iteratively solves the proximal problem and has to run on a series of lambda values. Actually, it does not really take much longer time to run on all the datasets than the other methods on a desktop with 4-core Intel i7-4790 CPU of 3.6 GHz, 32 GB RAM and windows 7 operating system. Our proposed method is implemented in Matlab.

To show that our proposed method can actually run efficiently, we pick the dataset "CMB" which is the largest dataset in terms of the number of samples and the resulting tree size, that we have used in our experiments. The tree to be pruned contains 4592 nodes. We varied the number of samples used in our proposed decision tree pruning algorithm from 3,000 to 10,000 and set r to be $\{2^{-8}, 2^{-9}, 2^{-10}, 2^{-11}\}$, respectively, and showed the corresponding execution time in Figure 3.7.

3.7 Conclusion

In this paper, we proposed a novel optimization formulation for the decision tree post-pruning problem. With the use of the indicator functions to represent a tree that was adopted in RuleFit, we are the first one to develop a systematic optimization formulation that can encapsulate the tree structure existing among the rules into a sparse learning framework by using the max-heap constraint as well as the sparsity constraint. This novel formulation leads to a non-convex optimization problem which

Table 3.2: Statistics of datasets used in the task of regression and performance of various methods. The classification performance is measured in terms of the Root Mean Squared Error (RMSE)

datasets	`# features`	`# samples`	REP(RMSE)	PEP(RMSE)	CCP(RMSE)	Maxheap(RMSE)
housing	13	506	**0.1686**	0.3424	0.1699	0.1691
parkinsonupds	16	5875	0.4163	0.4437	0.4165	**0.4054**
CMB	16	11934	0.1196	0.3445	0.1208	**0.1061**
skillcraft	18	3395	0.2814	0.4003	0.2830	**0.2754**
redwine quality	11	1599	0.2701	0.3052	0.2658	**0.2586**
whitewine quality	11	4898	0.2463	0.2829	0.2448	**0.2388**
bank data	32	8192	0.1667	0.2403	0.1680	**0.1647**
family data	32	8192	0.1673	0.2408	0.1690	**0.1655**
cpusmall	12	8192	0.0445	0.2194	0.0455	**0.0430**

Table 3.3: The average number of nodes in the decision trees pruned by different methods. Each column shows the average number of nodes in the tree after the original decision tree was pruned by the corresponding method. "N/A" means the method is not applicable to the task.

datasets	`Before pruning`	REP	PEP	CCP	MEP	Maxheap
sonar	17.2	5.6	11	8.2	9.6	6.3
biodeg	96.4	2	43.6	25.8	38.2	30.8
bank note	30.6.	10.2	21.4	23	24.8	13.5
Musk1	38.4	8	23.6	19.2	23.4	16.4
Musk2	168.4	1	101.2	87.4	89.0	57
EEGEyeState	1145.6	31.8	627.4	508	619	635.3
LSVT	9.6	3.2	6	4.2	5.2	3.8
masses	164	3.6	8.6	8.8	35.2	10.7
arcene	11.4	4.6	8.4	5.2	6.4	4.5
madelon	158	16.8	119.8	36.8	67.8	38.4
housing	310.2	70	1	30.6	N/A	40
parkinsonupds	2585.8	208.2	1	26	N/A	44
CMB	4559	2178	40.8	2878	N/A	3688.8
skillcraft	778.8.	120.2	1	43.2	N/A	38.2
redwine quality	303.2	55.6	1	16.2	N/A	54.3
whitewine quality	843.2	143.6	1	27	N/A	55.5
bank data	2974.6	127	1	35.4	N/A	44.7
family data	3031.4	134.4	1	38.6	N/A	38.2
cpusmall	2633.8	628	1	167.2	N/A	153.1

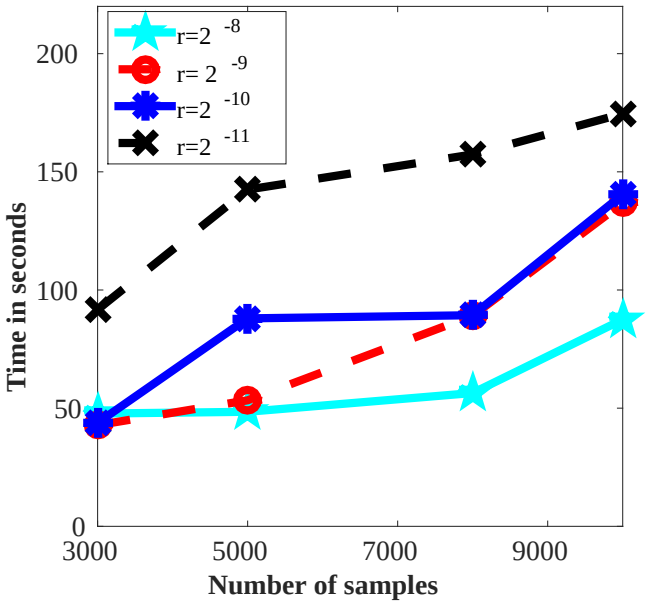


Figure 3.7: Pruning time on the tree generated from dataset "CMB". The tree has 4592 nodes. We varied the number of samples used for pruning the tree from 3,000 to 10,000 and chose r from $\{2^{-8}, 2^{-9}, 2^{-10}, 2^{-11}\}$.

can be addressed by an efficient max-heap projection algorithm that solves convex and smooth problems in each iteration. Since the selection of the regularization parameters has been a practical barrier for many sparse learning methods, an efficient stability selection method is further proposed for enabling robust model selection in practice. We conducted extensive experiments using a wide range of real-world datasets which demonstrated that the proposed method outperforms many existing benchmark pruning methods, leading to better prediction accuracy. Note that our proposed method provides a generic framework that establishes an interesting connection between the decision tree pruning with sparse learning, the same framework can be adopted to a wide range of decision tree learning and pruning scenarios for accommodating different situations and decision-making demands.

One of our future directions is to design a method that could simultaneously prune multiple decision trees with each pruner being aware of the structure of the other trees in the forest. This collective pruning method will not only reduce the size of each tree but also potentially yield a more compact forest. For instance, by imposing certain structure constraints on all the trees in the forest, rules that appear in one tree will less likely appear in another. Such a method would be of great interest since once a forest has been pruned, the computation time for generating a prediction would be reduced and the prediction accuracy could be improved.

Chapter 4

GRADIENT BOOSTING DECISION TREE PRUNING

4.1 Introduction

Although well-known for its advantages such as being a highly non-linear model, capable of dealing with heterogeneous data where features may come from different sources as well as being translatable to easy-to-interpret rules, a single decision tree model is not known to have the advantage of achieving the state-of-the-art predictive performance. One way that has been widely used in both academia and industry to compensate for the relatively poor predictive performance is to use tree ensemble which produces prediction by combining the output from multiple tree models.

For a given a dataset of n samples,$\mathcal{D} = \{(\mathbf{d_i}, y_i)\}$ where $i = 1, \cdots, n$ and $\mathbf{d_i}$ and y_i are the feature vector and label/response for the i-th sample. The tree ensemble model consists of M trees can be essentially written as

$$\hat{y}_i = \phi(\mathbf{d}_i) = \sum_{m=1}^{M} \alpha_m f_m(\mathbf{d}_i), \quad f_m \in \mathcal{F}$$

where $\mathcal{F}$ is the functional space of classification and regression tree. α_m is the weight associated with the m-th ensemble member.

Tree boosting, in which the ensemble members are regression trees, is a very successful example of a large category of boosting algorithms. During the past few decades, different types of boosting algorithms have been studied and explored from the perspectives of both theory and practice. The difference between these different boosting algorithms lies mainly in their hypothesis and choice of strategy in weighing the training samples. Among all the boosting algorithms, perhaps the most classic

and historically significant one is Adaboost Freund *et al.* (1999) which adapts well to the weak learners by employing a strategy that assigning a greater weight to those samples misclassified by previous ensemble members when each time a new learner is to be trained and added to the ensemble. The other type of boosting algorithm that draws an increasing amount of attention in recent years is gradient boosting since it not only has been demonstrated to be able to produce state-of-the-art results on classification, regression, ranking problems but also become the prototype of methods that has led to the winning of quite a few data mining and machine learning challenges.

In the original gradient boosting decision tree (GBDT) proposed by Friedman (2001), the authors consider that the optimal ensemble of the members in $\mathcal{F}$ ought to be the one that minimizes the expected loss which, when only a finite of samples are involved, can be written as

$$\phi^* = \arg\min_{\phi} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \phi(\mathbf{d_i}))$$

where L is the loss function. For a dataset with n samples, the entire functional space were treated as $\mathbb{R}^n$. In original gradient boosting decision tree, the base learners (ensemble members) are added to the ensemble one after another and only the first-order information was taken into consideration. The new base learner is chosen to be the one from $\mathcal{F}$ that is closest to the negative gradient of loss function with respect to ϕ evaluated in the functional space at the point given by the current estimation of the ensemble. The step size is determined by solving an optimization problem of a single one-dimensional variable that results in the greatest reduction in the loss function.

As is pointed out by Johnson, et al Johnson and Zhang (2014) and Zhang, et al Zhang *et al.* (2005)), although GBDT has been very successful in many applications due to its capability of identifying non-linear interactions between variables from potentially heterogeneous data, there are not without disadvantages which are mainly

manifested in the fact that there is no explicit regularization term. In practice, Usually only tree stumps (e.g. the depth of each tree is less than or equal to three) are generated as ensemble members to avoid overfitting. Also, it is argued in Zhang *et al.* (2005) that a shrinkage parameter can be added for each ensemble member and a early-stopping criterion could be potentially employed as a form of regularization. Yet the interaction between these parameters and its effect on the eventual ensemble model is unclear, which could result in the implicit regularization being not effective. In addition, A small shrinkage parameter could possibly lead to a large number of trees in the ensemble, which could lead to huge computational cost when it comes to making predictions. The other disadvantage of the original GBDT is that it only utilizes the first-order information when it generates a new ensemble member, making the process of incrementally adding regression trees potentially not so efficient.

In practice, all these disadvantages are translated into the need to pick a few hard-to-tune parameters such as the depth of each tree and the number of regression trees in the ensemble. Once these parameters are fixed, they impose a great constraint on the flexibility of the ensemble members (the scope of the functional space $\mathcal{F}$) as well as the form of the additive model could take, making the resulting tree ensemble more likely to either overfit or stop growing the forest too early.

In partially resolving these issues, some of more recently proposed tree boosting algorithms such as Regularized Greedy Forest (RGF) Johnson and Zhang (2014) and and Xgboost Chen and Guestrin (2016) have introduced an explicit regularization term to prevent the model from overfitting. One of the regularization terms defined in RGF, for instance, imposes higher penalty on nodes of great depth, thus reducing the structural risk of each tree by keeping it from growing too deep. By imposing an upper bound on the number of leaf nodes in the model, RGF does not place explicit constraint on the regression tree functional space or the number of ensemble

members. In Xgboost, with the number of leaf nodes being explicitly included in the regularization term, the growth of a tree automatically stops when reduction in empirical loss function brought about by further split of a leaf node cannot compensate for the increase of structural penalty. In addition to using an explicit regularization term, both methods have employed a greedy strategy in searching for a feature and the threshold on that feature to split a leaf node with the objective of reducing the loss function as much as possible. However, there is no guarantee that employing such a greedy strategy for splitting each leaf node is capable of producing an optimal regression tree or yielding an ensemble of stronger predictive power. A case in point is Random Forest which works pretty well by randomly selecting features to split nodes.

In this chapter, we employ the same decision tree pruning method as presented in the previous chapter to prune each ensemble member of GBDT immediately after it is generated. This new approach still keeps the spirit of gradient boosting in adding ensemble members but eliminates the need to specify the number of trees and maximum depth of each tree, making the training process more flexible and easily controllable with the addition of a single regularization parameter. Furthermore, to improve the efficiency of the model which might have the problem of redundancy as a result of the use of a small shrinkage parameter, we further introduce two approaches, GBDT compression which simultaneously prunes all the regression trees in GBDT, and GBDT reduction which simultaneously prunes and selects the ensemble members in GBDT. All these methods are tested for the task of prognosis of treatment of depression on a dataset collected from STAR*D trial, a clinical trial designed to determine the comparative effectiveness of different treatment options for patients with depression.

4.2 Post-Pruning GBDT

In adapting our proposed decision tree pruning algorithm for GBDT, we still keep the practice of incrementally adding trees to the ensemble to reduce the empirical error on the training samples while introducing a regularization term to control for the model complexity at the same time. Suppose we have $t-1$ trees already in the ensemble. Denoting the prediction given by these $t-1$ trees as $\hat{y}^{(t-1)}$, we add the t-th tree with the intention of minimizing the following objective function.

$$\min_{f_t} \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{d}_i)) + \Omega(f_t)$$

where $\Omega(\cdot)$ is the regularization function that imposes sparsity as well as Max-heap constraint to control for the complexity of the new tree.

Applying second-order Taylor expansion to the above objective function, we can rewrite the loss function as approximated by the following function :

$$\min_{f_t} \sum_{i=1}^{n} \left(L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(\mathbf{d}_i) + \frac{1}{2} h_i f_t^2(\mathbf{d}_i) \right) + \Omega(f_t) \tag{4.1}$$

$$= \min_{f_t} C^{(t-1)} + \sum_{i=1}^{n} \left(g_i f_t(\mathbf{d}_i) + \frac{1}{2} h_i f_t^2(\mathbf{d}_i) \right) + \Omega(f_t) \tag{4.2}$$

where $g_i = \frac{\partial L(y_i, \hat{y})}{\partial \hat{y}}|_{\hat{y}=\hat{y}_i^{(t-1)}}$ and $h_i = \frac{\partial^2 L(y_i, \hat{y})}{\partial \hat{y}^2}|_{\hat{y}=\hat{y}_i^{(t-1)}}$; $C^{(t-1)}$ is a constant that is independent of the new function (tree) f_t we are trying to add. In the following discussion, we simply drop this constant term for simplicity.

Although Regularized Greedy Forest Johnson and Zhang (2014) and XgboostChen and Guestrin (2016) use different regularization terms, they both adopt the same strategy of updating the structure of the forest (i.e. which node to split using which feature at what threshold) and the weights of the nodes alternatively and using the regularized loss function as a scoring function to determine how to update the structure of forest. Since it is impossible to enumerate all the possible tree structure all

at once, a greedy algorithm is employed to grow a tree (or a forest). In Regularized Greedy Forest, for instance, structure change is limited to either adding a new tree stump or selecting one of the most recently added T nodes to split. Which one to use depend on whichever option could lead to the greatest descent of the regularized loss function. In Xgboost, instead of enumerating all the structures that this new tree could possibly assume, it starts from a single leaf and iteratively chooses nodes to split based on the regularized loss function until further splits can no longer bring down the objective function.

Instead of adjusting the structure of the tree and updating the weights for the leaf nodes alternatively, we adopt a "post-pruning" approach where a regression tree is first trained based on the principle of minimizing the empirical loss of (4.2), which is followed by a pruning step attempting to minimize the entire regularized loss.

4.2.1 Train a Regression Tree

When the regularization term is not taken into consideration, the loss function (4.2) becomes

$$\sum_{i=1}^{n} \left(g_i f_t(\mathbf{d}_i) + \frac{1}{2} h_i f_t^2(\mathbf{d}_i) \right) \tag{4.3}$$

$$= \sum_{i=1}^{n} \frac{h_i}{2} \left(f_t(\mathbf{d}_i) + \frac{g_i}{h_i} \right)^2 - \sum_{i=1}^{n} \frac{g_i^2}{2h_i} \tag{4.4}$$

It is clear that without regularization term, the procedure is equivalent to training a regular regression tree with each sample $\mathbf{d}_i$ weighted by h_i which is the second order derivative of the loss function with respect to the current estimate and its target being $-\frac{g_i}{h_i}$.

4.2.2 Prune the Regression Tree

Suppose we have $p+1$ nodes in the new regression tree $v_0, \cdots, v_p$. Each node $v_j (0 \leq i \leq p)$ represents a rule function that maps all the n samples in the dataset $\mathbf{D}$ to a n-dimensional vector consisting of only 0 or 1 where whether a entry takes on 0 or 1 depends on whether the corresponding sample satisfies conjunction of the rules on the path connecting the root to node v_j. Let $\mathbf{x}_j = v_j(\mathbf{D})$. With these $p+1$ nodes and their corresponding rule function, we can obtain a new representation of the original dataset $\mathbf{D}$ which we denote as $X = [\mathbf{x}_p, \cdots, \mathbf{x}_p] = [\mathbf{s}_1^T; \cdots; \mathbf{s}_n^T]$.

Post-pruning decision tree algorithm involves the use of a bias b and a weight vector $\mathbf{w}$ where the j-th entry w_j corresponds to the weight associated with the node v_j with the final prediction for the i-th sample $\mathbf{d}_i$, which becomes $\mathbf{s}_i$ after mapping with all the rule functions, given by $\hat{y}_i = \mathbf{w}^T \mathbf{s}_i + b$. The pruning of decision tree is implemented via a procedure of learning the optimal $\mathbf{w}$ and b that imposes sparse constraint which enforces some entries of the weight $\mathbf{w}$ to be zeros as well as a max-heap constraint that requires the absolute value of the weight of a node be greater than that of its child nodes. By this max-heap constraint, if the weight of the node shrinks to zero, all its descendant nodes will have zero weight.

Denote the weight vector and bias for the t-th regression tree in the ensemble as $\mathbf{w}_t, b_t$, respectively. The prediction of response of the i-th sample is given by $\hat{y}_i = \mathbf{w}_t^T \mathbf{s}_i + b_t$, and regularization term for f_t is

$$\Omega(f_t) = \begin{cases} \lambda \|\mathbf{w}_t\|_1, & \text{if } \mathbf{w}_t \in P \\ +\infty, & \text{if } \mathbf{w} \notin P. \end{cases}$$

The regularized objective function (4.2) can be transformed into the following problem:

$$\min_{\substack{\mathbf{w}_t, b_t \\ \mathbf{w}_t \in P}} \quad \sum_{i=1}^{n} \left(g_i \left(\mathbf{w}_t^T \mathbf{s}_i + b_t \right) + \frac{1}{2} h_i \left(\mathbf{w}_t^T \mathbf{s}_i + b_t \right)^2 \right) + \lambda \|\mathbf{w}_t\|_1 \tag{4.5}$$

$$= \min_{\substack{\mathbf{w}_t, b_t \\ \mathbf{w}_t \in P}} \quad \sum_{i=1}^{n} \left(\frac{1}{2} h_i \mathbf{w}_t^T \mathbf{s}_i \mathbf{s}_i^T \mathbf{w}_t + (g_i + h_i b_t) \mathbf{s}_i^T \mathbf{w}_t + g_i b_t + \frac{1}{2} h_i b_t^2 \right) + \lambda \|\mathbf{w}_t\|_1 \tag{4.6}$$

$$= \min_{\substack{\mathbf{w}_t, b_t \\ \mathbf{w}_t \in P}} \quad \frac{1}{2} \mathbf{w}_t^T X^T B X \mathbf{w}_t + \left(\mathbf{g}^T + b_t \mathbf{1}_n^T B \right) X \mathbf{w}_t + \mathbf{1}_n^T \mathbf{g} b_t$$

$$+ \frac{1}{2} \mathbf{1}_n^T B \mathbf{1}_n b_t^2 + \lambda \|\mathbf{w}_t\|_1 \tag{4.7}$$

where $B = diag(h_1, \cdots, h_n)$; $\mathbf{g} = (g_1, \cdots, g_n)^T$ and $\mathbf{1}_n$ is a column vector consists of n entries of 1.

Let $L_t = \frac{1}{2} \mathbf{w}_t^T X^T B X \mathbf{w}_t + \left(\mathbf{g}^T + b_t \mathbf{1}_n^T B \right) X \mathbf{w}_t + \mathbf{1}_n^T \mathbf{g} b_t + \frac{1}{2} \mathbf{1}_n^T B \mathbf{1}_n b_t^2$, which is the empirical loss of (4.7).

we have

$$\frac{\partial L_t}{\partial \mathbf{w}_t} = X^T B X \mathbf{w}_t + X^T (\mathbf{g} + b_t \mathbf{1}_n); \tag{4.8}$$

$$\frac{\partial L_t}{\partial \mathbf{b}_t} = \mathbf{1}_n^T \mathbf{g} + \mathbf{1}_n^T B \mathbf{1}_n b_t + \mathbf{1}_n^T B X \mathbf{w}_t. \tag{4.9}$$

Plugging (4.8) and (4.9) into (3.6) and applying the Algorithm 3, we could obtain a pruned ensemble member. The detailed description of the algorithm is given in Algorithm 4.

4.3 GBDT Compression and Reduction

One of the observations made by Friedman regarding gradient boosting decision tree is that the shrinkage parameter may have to be small, preferably infinitesimal, in order to achieve good predictive performance in practice. Such an observation was further supported by the theoretical analysis in Zhang *et al.* (2005). In our experiments, we also observed a fairly similar phenomenon, that is, to certain extent,

Algorithm 4 GBDT pruning

Input: Dataset $\mathcal{D} = \{(\mathbf{d}_i, y_i)\}_{i=1}^n$, λ, s.

Output: A set of trees $\{T_j\}$ and the set of the corresponding weight vector and bias $\{(\mathbf{w}_j, b_j)\}$

$j \leftarrow 1$; $\hat{\mathbf{y}} \leftarrow 0$

while True **do**

  `Compute negative gradient` $\{-g_i\}_{i=1}^n$ `and second order coefficient` $\{h_i\}_{i=1}^n$ `based on current estimate` $\hat{y}_i$ `and label` y_i.

  $T_j,\ X_j \leftarrow$ `Train regression tree on` $\{\mathbf{d}_i, -g_i/h_i\}_{i=1}^n$.

  $\mathbf{w}_j, b_j \leftarrow$ `Prune regression tree on` $(X_j, -\mathbf{g}/\mathbf{h}, \lambda)$ using Algorithm 3.

  `Update current estimate` $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} + s * (X_j \mathbf{w}_j + b_j \mathbf{1}_n)$.

  if $\|\mathbf{w}_j\|_1$ `is less than certain tolerance level;` **then**

    `break`

  end if

  $j \leftarrow j + 1$

end while

predictive performance favors a small shrinkage parameter. However, as is argued in Johnson and Zhang (2014), an excessively small shrinkage parameter can lead to a huge model, which is undesirable due to the high memory and computation cost that it introduces for applications, especially when it comes to making predictions using the pre-trained model.

In this section, we propose two approaches aimed at addressing this issue. In the first subsection, we propose an approach that can simultaneously prune all the trees in the forest. In the second section, we propose a method can simultaneously prune all the trees in the forest and selecting trees up to a specified number. One thing we would like to drawing attention here is that we use $\boldsymbol{\beta}$ to denote the weight associated

with the nodes in the entire forest and $\boldsymbol{\beta}_i$ to denote the the weight associated with nodes in the i-th tree to highlight the difference from previous notation which uses $\mathbf{w}$ to denote the weight associated with nodes of a single tree.

4.3.1 GBDT Compression - Simultaneously Pruning Trees in GBDT

In this subsection. we consider the problem of how to simultaneously prune multiple decision trees of GBDT. Suppose that we have a forest which consists of K randomized trees $\{T_1, T_2, \cdots, T_K\}$. Let v_{ji} be the i-th node of the j-th tree and x_{ji} be the realization of the rule function corresponding to the node v_{ji} on all the samples. Let $X_j = \left[\mathbf{x}_{j0}, \mathbf{x}_{j1}, \cdots, \mathbf{x}_{j,p_j}\right]$ where $p_j + 1$ is the number of nodes in the tree T_j with the the weights for corresponding nodes in the tree T_j being the column vector $\boldsymbol{\beta_j} = \left[\beta_{j0}, \beta_{j1}, \cdots, \beta_{j,p_j}\right]$. The prediction is given by $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta} + b \cdot \mathbf{1}_n$. Our goal is to simultaneously prune all the decision trees in the forest such that the regularized loss function is minimized under the constraint that the absolute value of weights of all the nodes of each tree is subject to the max-heap constraint, which can be formulated as the following optimization problem.

$$\min_{\boldsymbol{\beta}, b} H_{(\mathbf{X},\mathbf{y})}(\boldsymbol{\beta}, b) + \lambda \|\boldsymbol{\beta}\|_1 \tag{4.10}$$

$$s.t. \quad \boldsymbol{\beta_j} \in P, \quad j = 1, \cdots, T \tag{4.11}$$

where $\mathbf{X} = [X_1, X_2, \cdots, X_T]$ (i.e. a horizontal concatenation of matrices X_j where $j = 1, 2, \cdots, T$), $\boldsymbol{\beta} = \left[\boldsymbol{\beta_1^T}, \cdots, \boldsymbol{\beta_T^T}\right]^T$. $H_{(\mathbf{X},\mathbf{y})}$ is the loss function defined on all the ensemble members in GBDT.

Note that the only difference between this problem and the problem solved in the case of single decision tree pruning is that instead of imposing the max-heap constraint on all the variables, the variables in this problem are divided into T groups with the max-heap constraint imposed upon each group. The GIST framework can

still be applied to solve this problem. In this case, the proximal regularization of H linearized at a given point $\left(\boldsymbol{\beta}^{(k)}, b^{(k)}\right)$ is

$$\begin{aligned}\left(\boldsymbol{\beta}^{(k+1)}, b^{(k+1)}\right) = &\arg\min_{\boldsymbol{\beta},b} \left\langle \nabla_{\boldsymbol{\beta}} H_{(\mathbf{X},\mathbf{y})}\left(\boldsymbol{\beta}^{(k)}, b^{(k)}\right), \boldsymbol{\beta} - \boldsymbol{\beta}^{(k)} \right\rangle \\ &+ \left\langle \nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\boldsymbol{\beta}^{(k)}, b^{(k)}\right), b - b^{(k)} \right\rangle + \frac{t^k}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)}\|_2^2 \\ &+ \frac{t^k}{2}\|b - b^{(k)}\|_2^2 + \sigma_1(\boldsymbol{\beta}) \end{aligned} \tag{4.12}$$

where

$$\sigma_1(\boldsymbol{\beta}) = \begin{cases} \lambda\|\boldsymbol{\beta}\|_1, & \text{if } \boldsymbol{\beta_j} \in P, \quad j = 1, \cdots, T \\ +\infty, & \text{otherwise.} \end{cases} \tag{4.13}$$

In the same way, we have

$$b^{(k+1)} = b^{(k)} - \frac{\nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\boldsymbol{\beta}^{(k)}, b^k\right)}{t^k} \tag{4.14}$$

$$\begin{aligned} \boldsymbol{\beta}^{(k+1)} = &\arg\min_{\boldsymbol{\beta}} \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\alpha}^{(k)}\right\|_2^2 + \frac{\lambda}{t^k}\|\boldsymbol{\beta}\|_1 \\ &s.t. \quad \boldsymbol{\beta_j} \in P \end{aligned} \tag{4.15}$$

where $\boldsymbol{\alpha}^{(k)} = \boldsymbol{\beta}^{(k)} - \frac{\nabla_{\boldsymbol{\beta}} H_{(\mathbf{X},\mathbf{y})}\left(\boldsymbol{\beta}^{(k)}, b^{(k)}\right)}{t^k}$. Apparently, the problem (4.15) can be decomposed to T problems formulated in (3.7).

4.3.2 GBDT Reduction - Simultaneously Pruning and Selecting Trees in GBDT

In this subsection, we consider the problem of simultaneous selecting decision trees and pruning selected decision trees. The idea of simultaneous selecting decision trees and pruning selected decision trees is illustrated in Figure 4.1.

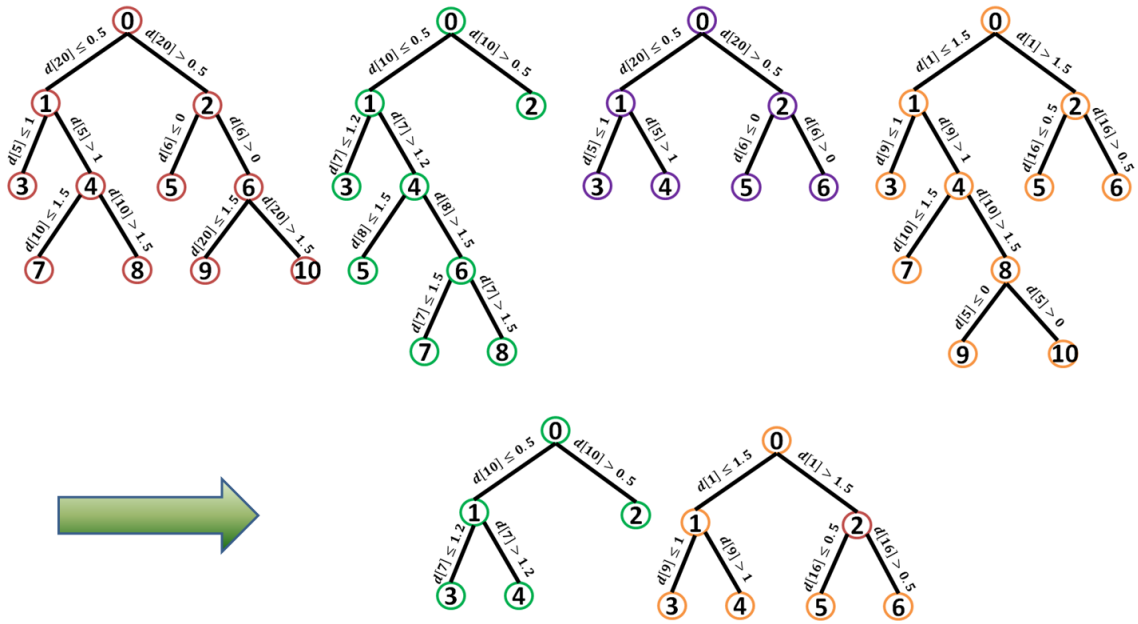


Figure 4.1: Simultaneously pruning and selecting trees in GBDT

In addition to associating the i-th node in the tree T_j with a weight β_{ji} indicating the importance of the node in determining labels, we introduce another parameter r which is the upper bound of the number of trees to be selected. We implicitly assume that r is less than the number of trees in the original GBDT. otherwise it would simply degenerate into the case discussed in the previous subsection. Following notation we used in the previous sections, we still associate the vector $\boldsymbol{\beta}_j$ with the tree T_j as the weights for the nodes in the tree. With the number of selected trees to be less than r, we can formulate the proposed idea into the following optimization problem.

$$\begin{aligned} &\min_{\boldsymbol{\beta},b} H_{(\mathbf{X},\mathbf{y})}\left(\boldsymbol{\beta},b\right)+\lambda\|\boldsymbol{\beta}\|_1 \\ &s.t. \quad \boldsymbol{\beta_j}\in P_j, \quad j=1,\cdots,T \\ &\qquad \sum_{j=1}^{T} I\left(\|\boldsymbol{\beta}_j\|_1\neq 0\right)\leq r \end{aligned} \tag{4.16}$$

where $I\left(\cdot\right)$ is the indicator function.

Similar to the way that the problem is solved in the previous subsection, we can define the proximal regularization of H linearized at a given point $\left(\boldsymbol{\beta}^{(k)}, b^{(k)}\right)$ to be

$$\begin{aligned}\left(\boldsymbol{\beta}^{(k+1)}, b^{(k+1)}\right) = & \arg\min_{\boldsymbol{\beta}, b} \left\langle \nabla_{\boldsymbol{\beta}} H_{(\mathbf{X},\mathbf{y})}\left(\boldsymbol{\beta}^{(k)}, b^{(k)}\right), \boldsymbol{\beta} - \boldsymbol{\beta}^{(k)} \right\rangle \\ & + \left\langle \nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\boldsymbol{\beta}^{(k)}, b^{(k)}\right), b - b^{(k)} \right\rangle \\ & + \frac{t^k}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)}\|_2^2 + \frac{t^k}{2}\|b - b^{(k)}\|_2^2 + \sigma_2(\boldsymbol{\beta}) \end{aligned} \tag{4.17}$$

where

$$\sigma_2(\boldsymbol{\beta}) = \begin{cases} \lambda\|\boldsymbol{\beta}\|_1, & \text{if } \boldsymbol{\beta}_j \in P_j, \quad j = 1, \cdots, T \text{ and } \sum_{j=1}^{T} I\left(\|\boldsymbol{\beta}_j\|_1 \neq 0\right) \leq r \\ +\infty, & \text{otherwise.} \end{cases} \tag{4.18}$$

In the same way, we have

$$b^{(k+1)} = b^{(k)} - \frac{\nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\boldsymbol{\beta}^{(k)}, b^k\right)}{t^k} \tag{4.19}$$

$$\begin{aligned} \boldsymbol{\beta}^{(k+1)} = & \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\alpha}^{(k)}\right\|_2^2 + \frac{\lambda}{t^k}\|\boldsymbol{\beta}\|_1 \\ s.t. \quad & \boldsymbol{\beta_j} \in P_j \quad j = 1, \cdots, T \\ & \sum_{j=1}^{T} I\left(\|\boldsymbol{\beta}_j\|_1 \neq 0\right) \leq r \end{aligned} \tag{4.20}$$

where $\boldsymbol{\alpha}^{(k)} = \boldsymbol{\beta}^{(k)} - \frac{\nabla_{\boldsymbol{\beta}} H_{(\mathbf{X},\mathbf{y})}\left(\boldsymbol{\beta}^{(k)}, b^{(k)}\right)}{t^k}$. Similar to the problem (4.15), the problem (4.20) can be decomposed to T independent problems formulated in (3.7), with the only difference manifested in the additional constraint that no more than r groups of variables can be selected.

Since the optimal solutions to the T subproblems are completely independent of one another, we can first solve all of the T subproblems and then adopt a greedy strategy to deal with the constraint of allowing no more than r groups of variables to be selected. Suppose that we have solved all these T independent subproblems

implicated in problem (4.20) with $\boldsymbol{\beta}_i^* = \arg\min_{\boldsymbol{\beta}_i} \frac{1}{2} \left\| \boldsymbol{\beta}_i - \boldsymbol{\alpha}_i^{(k)} \right\|_2^2 + \frac{\lambda}{t^k} \|\boldsymbol{\beta}_i\|_1$ s.t. $\boldsymbol{\beta}_i \in P_i$ and $d_i = \frac{1}{2} \left\| \boldsymbol{\beta}_i^* - \boldsymbol{\alpha}_i^{(k)} \right\|_2^2 + \frac{\lambda}{t^k} \|\boldsymbol{\beta}_i^*\|_1$. Let $c_i = \frac{1}{2} \|\boldsymbol{\alpha}_i^{(k)}\|_2^2$. At the beginning when none of the $\boldsymbol{\beta}_i^*, i = 1, \cdots, T$ were selected, the value of the objective function $s_0 = \sum_{i=1}^{T} c_i$. At the first step, suppose that we select the j-th group of variable $\boldsymbol{\beta}_j^*$. The value of the objective function becomes

$$\begin{aligned} s_1 &= s_0 - c_i + d_i \\ &= s_0 - (c_i - d_i) \end{aligned} \tag{4.21}$$

In order to achieve maximum reduction in the objective function value, we wish the chosen group of variables $\boldsymbol{\beta}_j^*$ to have two effects: (1) its being chosen gives rise to reduction in objective function value, i.e. $c_i - d_i > 0$; (2) the objective function value it brings down should be maximum among all unselected groups of variables, i.e. $i = \arg\max_{j \in \mathbb{S}} c_i - d_i$ where $\mathbb{S} \subseteq \{1, \cdots, T\}$ and denotes the set of indices of groups that have not yet been selected. We can describe this idea to solve problem (4.20) using the Algorithm 5.

4.4 Data and Experiments

*4.4.1 STAR*D: an Introduction to Data and Tasks*

STAR*D [1] , which is the abbreviation for Sequenced Treatment Alternatives to Relieve Depression, is a clinical research initiative funded by National Institute of Mental Health (NIMH) focusing on evaluating the effectiveness of different treatment strategies and a combination of different treatment strategies on patients with non-psychotic major depression. The study spanned seven-years and enrolled a total of 4,041 patients from a broad spectrum of social demographics in United States.

[1] http://www.nimh.nih.gov/funding/clinical-research/practical/stard/index.shtml

Algorithm 5 Algorithm to solve problem (4.20)

Initialization:

$\boldsymbol{\alpha}^{(k)}, \lambda, t_k$, the max-heap constraint of T group of variables $\{P_j | j = 1, \cdots, T\}$, the maximum number of groups to be selected r.

1: **for** $i = 1$ to T **do**

2: $\quad \boldsymbol{\beta}_i^* = \arg\min_{\boldsymbol{\beta}_i} \frac{1}{2} \left\| \boldsymbol{\beta}_i - \boldsymbol{\alpha}_i^{(k)} \right\|_2^2 + \frac{\lambda}{t^k} \left\| \boldsymbol{\beta}_i \right\|_1$ s.t. $\boldsymbol{\beta}_i \in P_i$.

3: $\quad d_i = \frac{1}{2} \left\| \boldsymbol{\beta}_i^* - \boldsymbol{\alpha}_i^{(k)} \right\|_2^2 + \frac{\lambda}{t^k} \left\| \boldsymbol{\beta}_i^* \right\|_1$.

4: $\quad c_i = \frac{1}{2} \| \boldsymbol{\alpha}_i^{(k)} \|_2^2$.

5: **end for**

6: $\mathbf{x} \leftarrow \mathbf{0}$ with length equal to $\boldsymbol{\alpha}^{(k)}$.

7: $\mathbb{S} \leftarrow \{1, \cdots, T\}$.

8: **for** $i = 1$ to T **do**

9: $\quad i \leftarrow \arg\max_{i \in \mathbb{S}} c_i - d_i$

10: $\quad$ **if** $c_i > d_i$ **then**

11: $\quad\quad \mathbb{S} \leftarrow \mathbb{S} - i$.

12: $\quad\quad \mathbf{x}_i \leftarrow \boldsymbol{\beta}_i^*$.

13: $\quad$ **else**

14: $\quad\quad$ break;

15: $\quad$ **end if**

16: **end for**

Output: $\mathbf{x}$.

The study consists of a total of four sequential acute treatment stages. The patients that did not achieve symptom remission from one treatment stage could choose to go to the next stage or drop out from the study. Those patients who achieved symptom remission were encouraged to participate in a twelve-month naturalistic

follow-up phase in which their depressive symptom severity were continued to be measured on a monthly basis. Each acute treatment stage lasted somewhere between twelve weeks to fourteen weeks and were designed and carried out with the goal of attaining symptom remission. In each treatment stage, the patients' depressive symptom severity was measured every two to three weeks based on Quick Inventory of Depressive Symptomatology (QIDS) rated by both clinicians (QIDS-C) and patients' self-report (QIDS-SR) and the medication(s) under trial were dosed vigorously and provided for a sufficient duration to the extent that there was no report of intolerable side effects from the patients. All the patients were started on citalopram in the first treatment stage. The patients that did not achieve satisfactory treatment effects, upon obtaining their consent, was enrolled in the second treatment stage where they were randomized in to one of the seven treatment strategies.

The data collected from the study consist of a wide range of information concerning the patients including depressive symptom severity measured on QIDS scale, depressive symptom severity measured on 17-item Hamilton Rating Scale of Depression (HRSD), depressive symptom severity measured on 30-item Inventory of Depressive Symptomatology (IDS), Frequency and Intensity of Side Effects Rating (FISER), Quality of Life Enjoyment and Satisfaction Questionnaire (Q-LES-Q), Work and Social Adjustment Scale (WSAS), Work Productivity and Activity Impairment Questionnaire (WPAI), social demographics, general health comorbidities, psychiatric comorbidities, psychiatric history, medication history, etc. Of all these information, only depressive symptom severity measured on QIDS scale (QIDS-C and QIDS-SR) were sampled every two to three weeks. However, patient might not come to the assigned clinic for a scheduled regular clinic visit, causing intermittent missing values or drop out of the study which could due to a variety of reasons such as having already achieved remission and started on the twelve-month follow-up phase, or having

to quit the treatment and because of intolerable side effects and move onto the next treatment stage, or exiting the study all together for reasons unknown to research coordinators.

In addition to the presence of different types of missingness due to patients' attrition or missing certain scheduled clinic visits or phone interviews, the fact that different types of information are collected at different frequencies also complicates the data preprocessing and analysis. For instance, although patients' depressive symptom severity measured by QIDS-C and QIDS-SR were sampled every two or three weeks at each acute treatment stage, patients' depressive symptom severity measured by HDRS and IDS, along with other information concerning patients' daily function such as WSAS as well as quality of life such as Q-LES-Q, etc. are only collected at baseline and week 12 of each treatment stage. Furthermore, the block-wise missing pattern caused by different patients experiencing different number of treatment stages (i.e. in general, more information regarding patients' response to treatment and daily function is available for those experience relatively more treatment stages in comparison to those experience relatively fewer number of treatment stages.) also poses new challenges and calls for an ad-hoc solution in regarding to how to make full use of information collected from multiple treatment stages with some patients experiencing more treatment stages while others experiencing less. A diagram illustrating the data collected in STAR*D trials is shown in Figure 4.2.

The assessment of a patient's treatment response was made based on QIDS. Although during the acute treatment stages, the clinical decisions such as those of whether a patient has achieved remission or not and how much more or less doses of medication under trial should be prescribed to patients were made using the clinician-rated QIDS (QIDS-C), in a report Rush *et al.* (2006) published by the researchers who designed and implemented these clinical trials, self-rated QIDS (QIDS-SR) was

used as the primary measure to define outcomes for acute and follow-up phases instead due to the concern of undesirable bias that might be introduced in the ratings given by clinicians. Besides, all the symptom ratings of symptoms recorded during the follow-up phase were based on patients' self-evaluation and were collected through Interactive Voice Response (IVR).

In terms of determining the effectiveness of treatment strategies, Rush *et al.* (2006) also gives two different criteria: remission and response. In Rush *et al.* (2006), remission is defined as the QIDS-SR total score being less than or equal to five, which means there is no self-reported depressive symptom. Response is defined as at least a 50% reduction in QIDS-SR score from the baseline measurement.

In our analysis, we use both remission and response as the criteria to determine whether the trial was effective on each of the patients. Also, we use measurements from both QIDS-C and QIDS-SR to determine whether a patient has achieved remission or response, thus resulting in a cross-tabulation between remission, non-remission, response, non-response and QIDS-C and QIDS-SR.

In terms of tasks, we are interested in predicting whether a patient is able to achieve remission/response by the end of second treatment stage by using only the data collected from enrollment (including patients' information on social demographic, medical history, general health and psychiatric comorbidities, etc.), week 0 and week 2 of the first treatment stage. The number of patients that have been able to achieve remission and response by the end of the second treatment stage and those who have failed to do so are shown in Table 4.1 and Table 4.2. The number of patients involved in the third and fourth treatment stage were too small to allow for any statistically significant analysis.

Table 4.1: The number of remission and non-remission by the end of second treatment stage based on QIDS-SR and QIDS-C

	Remission	Non-remission	Total
QIDS-C	1812	642	2454
QIDS-SR	1635	667	2302

Table 4.2: The number of response and non-response by the end of second treatment stage based on QIDS-SR and QIDS-C

	Response	Non-response	Total
QIDS-C	2267	515	2782
QIDS-SR	2036	550	2586

4.4.2 Experimental Setup and Results

In our analysis, we use both symptom remission and treatment response as defined in the previous subsection obtained from ratings based on both QIDS-C and QIDS-SR by the end of the second acute treatment step to measure the effectiveness of treatment tested in trial.

Considering there is class-imbalance in each of the four cases, two thirds of the samples from the minority class and an equal number of samples from the majority class were randomly selected for the purpose of training the models and the rest of the samples were used for testing. This random split between training and testing set was carried out ten times and the mean performance is reported. The baseline methods used in our experiments for comparison include GBDT implemented in scikit-learn Pedregosa *et al.* (2011), Regularized Greedy Forest (RGF) Johnson and Zhang (2014) and Xgboost Chen and Guestrin (2016). The parameters in each model were deter-

Table 4.3: Performance of different methods on prognosis of treatment.

	QIDS-C		**QIDS-SR**	
Method	Remission	Response	Remission	Response
GBDT	0.771 ± 0.012	0.735 ± 0.013	0.782 ± 0.011	0.732 ± 0.009
RGF	0.752 ± 0.018	0.724 ± 0.015	0.759 ±0.012	0.717 ± 0.011
Xgboost	0.745 ± 0.015	0.711 ± 0.014	0.760 ± 0.012	0.712 ± 0.014
Prune	**0.777 ± 0.012**	**0.742 ± 0.012**	**0.791 ± 0.011**	**0.745 ± 0.009**
Compress	0.771 ± 0.012	0.729 ± 0.012	0.785 ± 0.008	0.737 ± 0.012
Reduction	0.771 ± 0.012	0.732 ± 0.012	0.786 ± 0.009	0.737 ± 0.011

mined by three fold cross-validation. Following the practice form Perlis (2013), the performance were evaluated by area under curve (AUC). The mean and standard deviation of AUC values from ten random split of training and testing set is reported. It is worthing noting here that although the test sets are imbalanced, the trained models running on these tests are not biased toward any class because training sets on which the models were built are balanced.

4.4.3 Conclusion

In this chapter, we extended the method for post-pruning a single decision tree proposed in the previous chapter to deal with pruning of GBDT. Specifically, a new regression tree that minimizes the the second-order expansion of the original loss function at the current estimate of the ensemble is trained and then post-pruned through learning an optimal weight vector for the nodes that satisfies max-heap constraint as well as minimizes an objective function taking the form of an composition of second-order expansion of the original loss function and l_1-norm of the weight vector.

In comparison with GBDT, it introduces a regularization parameter but no longer requires the number of ensemble members to be specified beforehand or a fixed maximum depth of each regression tree which, in the case of GBDT, essentially has to be small in order to obtain good predictive performance. Furthermore, in order to address the issues of redundancy caused by a small shrinkage parameter and the heavy computation cost that occurs when it comes to making predictions with a redundant model, we further proposed two approaches, one aimed at compressing the model by simultaneously pruning all the regression trees in the ensemble and the other aimed at reducing the model by simultaneously selecting and pruning the ensemble members. Our methods outperforms the baseline methods which includes GBDT, RGF and Xgboost on the tasks of predicting the symptom remission and treatment response based on STAR*D dataset.

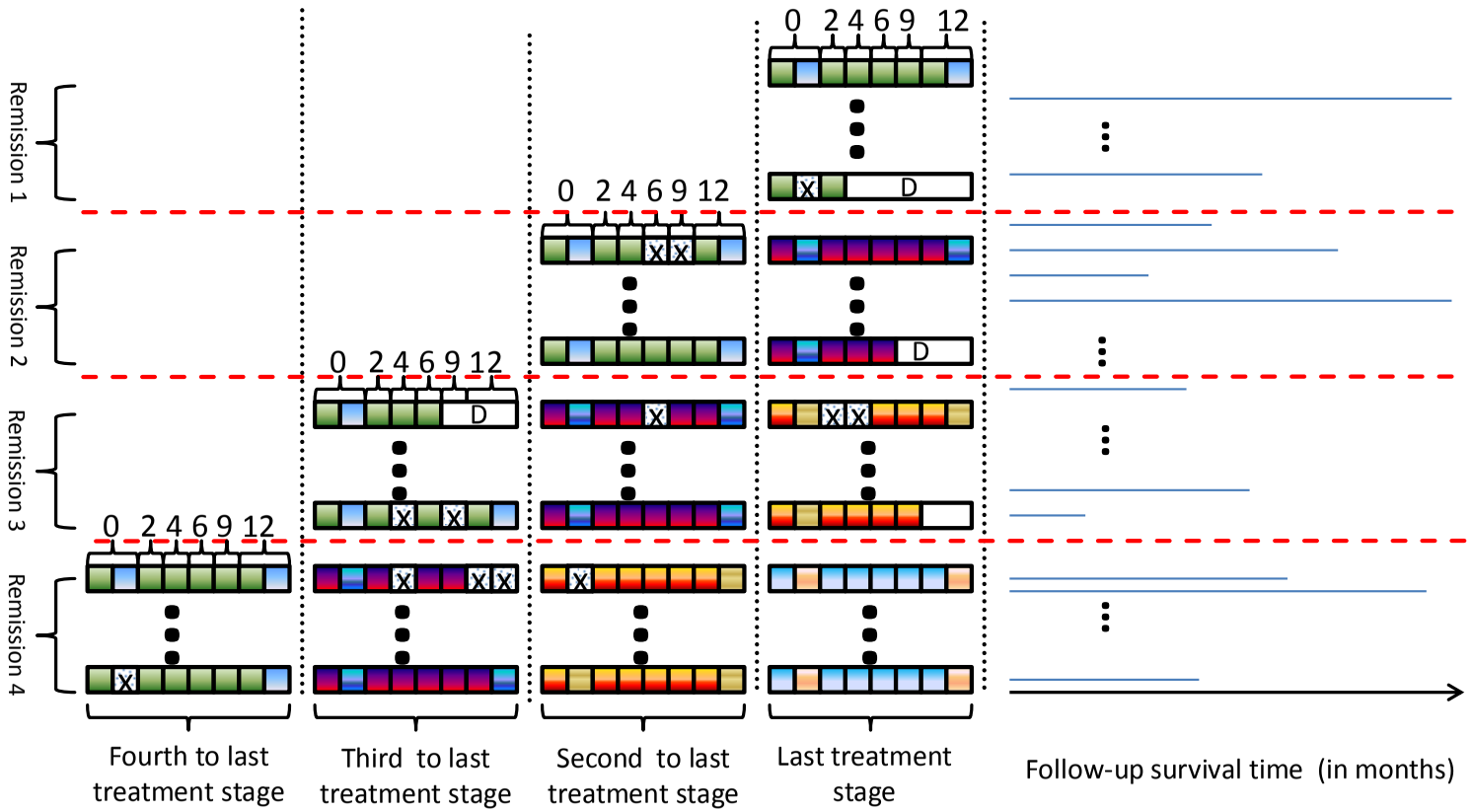


Figure 4.2: The diagram for data from STAR*D trial. The data collected from the same treatment stage is shaded with the same color. Data regarding patients' symptom severity are collected at baseline, week 2,4, 6, 9, 12 of each treatment stage. Some extra information is also available at baseline and week 12. The missing values caused by patients missing certain scheduled visits are marked by 'X' and 'D' denotes the missing values caused by patients dropping out of the study.

Chapter 5

PREDICT RISK OF RELAPSE FOR PATIENTS WITH MULTIPLE STAGES OF TREATMENT OF DEPRESSION

5.1 Introduction

Depression, clinically called Major Depressive Disorder (MDD), is a mood disorder that affects about one eighth of population in US Gaynes *et al.* (2009) and an estimated 350 million people globally and is projected on track to be the second leading cause of disability in the world by the year 2020 Marcus *et al.* (2012). Medications of different types, such as selective serotonin reuptake inhibitors (SSRIs) and serotonin norepinephrine reuptake inhibitors (SNRIs), which are based on different biological mechanisms and have different effects on neural activity have been developed and tested in a number of clinical trials during the past few decades. Typically, a clinical trial on one medication for depression lasts somewhere from one to four months during which the antidepressant under study is vigorously dosed to tolerance with the goal of symptom remission due to its implications of better daily function Rush *et al.* (2006)Kelsey (2004). However, the interplay of multiple factors such as patients' gene expression profile Eyre *et al.* (2015), chronicity of depression Alpert and Fava (2014), psychiatric and general heath comorbidities Otte (2008), intolerable side effects from medications, etc., makes many patients with MDD unlikely to respond to certain types of treatment. Thus they are unlikely to achieve remission with a single trial. For these patients, several sequential treatment stages are often necessary to obtain remission.

Another problem concerning the treatment of MDD is the potential risk of relapse among those who indeed achieve remission with one or several stages of treatment. Keller et al. MB *et al.* (1983) pointed out that there is a substantial probability of prompt relapse among patients without bipolar disorders who recovered from their first major depressive episode and should they relapse, they have an approximately 20% chance of remaining chronically depressed. Findings from Nierenberg *et al.* (2010) indicated that patients who achieved remission of MDD after treatment with citalopram still continued to experience residual symptoms which put them at a higher risk of relapse in a 12-month follow-up phase. Interestingly, the risk of relapse seems to be inversely proportional to the survival time after remission. For instance, Ramana et al. Ramana *et al.* (1995) found that all the relapses of subjects that participated in their study occurred within 10 months after they achieved remission. According to the results reported in MB *et al.* (1982), relapse occurs within four weeks for 12% of patients with remission. And it takes eight more weeks for the number to double. Thus it is of crucial importance to accurately predict the risk of relapse of MDD patients after their remission, especially those who are likely to relapse shortly after their remission as there is evidence showing that putting remitted patients under continuation and maintenance therapy would greatly reduce their risk of relapse Teasdale *et al.* (2000) Paykel (2001).

An example of clinical trial to treat depression involving both multiple stages of treatment aimed at achieving symptom remission and a follow-up phase evaluating the long-term treatment outcome is the Sequenced Treatment Alternatives to Relieve Depression (STAR*D) trial. It involves over 4,000 outpatients with nonpsychotic MDD from a broad spectrum of social demography. It consists of four sequential acute treatment stages. Patients who did not achieve remission or suffered from intolerable side effects of medications in one treatment stage were encouraged to go to

the next stage. Those who had achieved remission or shown significant symptomatic improvement could enter a 12-month naturalistic follow-up phase. In the follow-up phase, the measurements concerning patients' depressive symptoms were made on a monthly basis. However, in both acute treatment stages and follow-up, patients may miss certain scheduled clinic visits or drop out, resulting in different patterns of missingness.

One class of the widely used methods that can be potentially employed for building models to predict the risk of relapse is survival analysis, e.g., Cox proportional hazards model Cox (1972). However, traditional methods for survival analysis usually assume that for uncensored cases (e.g., subjects that relapsed), the exact time when the event of interest (e.g., relapse) occurs is known, which is not the case in STAR*D or for data collected through clinical trials of depression in general. In STAR*D, for instance, if a subject does not relapse at the 5th month but is observed to have relapsed at the 6th month, we only know for certainty that the relapse occurs somewhere between the 5th month and the 6th month. Another issue with traditional survival analysis methods is that they apply only to the situation where all the subjects have the same covariates. They cannot be readily used for solving the problem in which certain patients have covariates from more treatment stages while others have less. An alternative is to build a sequence of dependent predictive models, one model for each time point with the dependency formulated through an explicit or implicit constraint that if a model built for a later time point predicts that the event of interest does not occur for a subject, the models built for all the earlier time points should predict the same. However, such methods usually result in very complicated optimization problems to solve or have to resort to some kind of approximation Yu *et al.* (2011).

In this chapter, a censored regression approach is presented to predict the risk of relapse based on information collected from patients' acute treatment stages and

their enrollment. Specifically, we employ a truncated l_1 loss to model the responses (patients' time of relapse) that are upper bounded and/or lower bounded. Based on this basic loss function, we consider the hypothesis that can be represented as (1) an ensemble of decision trees, and (2) a linear combination of covariates. For the hypothesis that takes the form of an ensemble of decision trees, we develop a gradient boosting approach to learn the base models and combination coefficients. When the hypothesis takes the form of a linear combination of covariates, we develop a stochastic dual coordinate ascent algorithm, which is the state-of-the-art method for solving large-scale machine learning problems with a convex loss function Tran *et al.* (2015) with a guaranteed fast convergence rate Shalev-Shwartz and Zhang (2013). Furthermore, we assume that the treatment stage varying covariates collected on patients around the same time before they entered follow-up study should share some commonalities in terms of their relative contribution to predicted risk of relapse and, at the same time covariates collected from different stages overall, contribute differently to the prediction. Based on this assumption, we propose a multi-stage linear approach that can simultaneously estimate multiple linear models for patients remitted after different numbers of treatment stages. Based on data collected from STAR*D trial, we generated several datasets by selecting different cut-off points. Our results show that our proposed methods consistently outperform the Cox model on all datasets.

Major contributions of this chapter are summarized as follows: (1) We present a truncated l_1 loss based censored regression approach to deal with uncertainties of responses. (2) We develop an efficient gradient boosting algorithm and stochastic dual coordinate descent algorithm to solve the proposed formulation. (3) Based on the linear model, we further propose a multi-stage linear approach that can deal with covariates collected from different numbers of treatment stages. (4) We conduct

experiments on both synthetic datasets and STAR*D to evaluate the effectiveness of our methods. (5) We identify risk factors which might provide some new insights into development of more effective therapies for prevention of relapse.

5.2 Truncated l_1 Loss for Learning the Relapse time of Patients

In recent literature, different types of loss functions have been proposed to handle censored data under different application scenarios. For instance, in Khosla *et al.* (2010), the prediction of time of occurrence of stroke among subjects was modeled through the Huber loss which basically enforces the predicted time of the uncensored subjects to be the same as the observed time. In Shivaswamy *et al.* (2007), the loss function for learning from censored targets was absolute deviation of predicted value away from its target interval. In our situation, the assessments of depressive status of all the patients were made on a monthly basis during the follow-up phase, which means that when a patient was regarded as having relapsed at the time of assessment, we only know for sure that relapse of the patient occurred at or before the time when the assessment was made. For this reason, the truncated l_1 loss, which includes the loss function used in Shivaswamy *et al.* (2007) as a special case, is proposed to model the relapse risk of patients in the STAR*D cohort.

Suppose there are a total of n samples $D = \{x_1, \cdots, x_n\}$. Let $\mathcal{S}_l$ be the set of indices of samples whose responses are bounded from below by some values, i.e. $\mathcal{S}_l = \{i|a_i \geq l_i\}$ and $\mathcal{S}_u$ be the set of indices of samples whose responses are bounded from above by some values, i.e. $\mathcal{S}_u = \{i|a_i \leq u_i\}$ where a_i is the real unknown response of the i-th sample; l_i is its observed lower bound and u_i is its observed upper bound. The real unknown responses are both upper bounded and lower bounded for uncensored cases and only lower bounded for those censored cases.

We formulate our censored regression with truncated l_1 loss as the following optimization problem:

$$\begin{aligned}
&\arg\min_{F(\mathbf{x})} L\left(F(\mathbf{x})|\tau, \mathcal{S}_l, \mathcal{S}_u, \mathbf{l}, \mathbf{u}, D\right) \\
=&\arg\min_{F(\mathbf{x})} \left[\frac{\tau}{n}\sum_{i\in\mathcal{S}_l}\left(l_i - F(\mathbf{x}_i)\right)_+ + \frac{1-\tau}{n}\sum_{i\in\mathcal{S}_u}\left(F(\mathbf{x}_i) - u_i\right)_+\right], \qquad (5.1)
\end{aligned}$$

where $(z)_+ = \max(0, z), \forall z \in \mathbb{R}$; $F(\mathbf{x}_i)$ gives an estimate of the response for the i-th sample $\mathbf{x}_i$; $\mathbf{l}$, $\mathbf{u}$ are vectors comprising of the lower and upper bounds for all the samples, respectively; $\tau \in (0, 1)$ is a pre-specified constant that balances the trade-off between the two terms.

Intuitively, if the response of an instance $\mathbf{x}_i$ is lower bounded by l_i, we would like the predicted response $F(\mathbf{x}_i)$ to be greater than or equal to l_i. Otherwise, a penalty would be imposed. This works similarly if its response is upper bounded. No penalty is incurred if $F(\mathbf{x}_i) \in [l_i, u_i]$.

Although in practice, the response of an instance $\mathbf{x}_i$ is both lower bounded and upper bounded if it is an uncensored case, in the following, to simplify our discussion, we simply replicate such instances with one for lower bound index set $\mathcal{S}_l$ and one for upper bound index set $\mathcal{S}_u$ such that $\mathcal{S}_l \cap \mathcal{S}_u = \emptyset$ and denote the new dataset as X. Furthermore, we use one vector $\mathbf{c}$ to denote the union of $\mathbf{l}$ and $\mathbf{u}$ such that $c_i = l_i$ if $i \in \mathcal{S}_l$ and $c_i = u_i$ if $i \in \mathcal{S}_u$. Thus, the resulting loss can be written as

$$\begin{aligned}
&\arg\min_{F(\mathbf{x})} L\left(F(\mathbf{x})|\tau, \mathbf{c}, \mathcal{S}_l, \mathcal{S}_u, X\right) \\
=&\arg\min_{F(\mathbf{x})} \left[\frac{\tau}{N}\sum_{i\in\mathcal{S}_l}\left(c_i - F(\mathbf{x}_i)\right)_+ + \frac{1-\tau}{N}\sum_{i\in\mathcal{S}_u}\left(F(\mathbf{x}_i) - c_i\right)_+\right], \qquad (5.2)
\end{aligned}$$

where $N = |\mathcal{S}_l| + |\mathcal{S}_u|$. Note that replicating instances serves only to decouple the index set $\mathcal{S}_l$ and $\mathcal{S}_u$ and has no effect on model $F(\mathbf{x})$ trained on the dataset.

To further simplify the problem (5.2), we introduce variables y_i's ($i = 1, 2, \cdots, N$) which are defined as follows:

$$y_i = \begin{cases} 1 & \text{if } i \in \mathcal{S}_l \ ; \\ -1 & \text{if } i \in \mathcal{S}_u. \end{cases} \tag{5.3}$$

Then the problem (5.2) could be reformulated as

$$\begin{aligned} &\arg\min_{F(\mathbf{x})} L\left(F(\mathbf{x})|\tau, \mathbf{c}, \mathcal{S}_l, \mathcal{S}_u, X\right) \\ =&\arg\min_{F(\mathbf{x})} \tau \sum_{i\in\mathcal{S}_l} \left(c_i - F(\mathbf{x}_i)\right)_+ + (1-\tau) \sum_{i\in\mathcal{S}_u} \left(F(\mathbf{x}_i) - c_i\right)_+ \\ =&\arg\min_{F(\mathbf{x})} \tau \sum_{i\in\mathcal{S}_l} \left[y_i \left(c_i - F(\mathbf{x}_i)\right)\right]_+ \\ &+ (1-\tau) \sum_{i\in\mathcal{S}_u} \left[y_i \left(c_i - F(\mathbf{x}_i)\right)\right]_+ . \end{aligned} \tag{5.4}$$

Defining $\hat{c}_i, \hat{y}_i$ as

$$\hat{c}_i \ = \ \begin{cases} \tau y_i c_i & \text{if } i \in \mathcal{S}_l; \\ (1-\tau) y_i c_i & \text{if } i \in \mathcal{S}_u; \end{cases} \tag{5.5}$$

$$\hat{y}_i \ = \ \begin{cases} \tau y_i & \text{if } i \in \mathcal{S}_l; \\ (1-\tau) y_i & \text{if } i \in \mathcal{S}_u, \end{cases} \tag{5.6}$$

we can rewrite the problem (5.4) as

$$\arg\min_{F(\mathbf{x})} \sum_{i=1}^{N} \left(\hat{c}_i - \hat{y}_i F(\mathbf{x}_i)\right)_+ . \tag{5.7}$$

5.3 A Gradient Boosting Approach to Censored Regression with Truncated l_1 Loss

In this section, we consider the case where $F(\mathbf{x})$ can be represented as an ensemble of regression trees. Namely, for an ensemble consisting of $M+1$ base learners:

$$F(\mathbf{x}) = \sum_{i=0}^{M} \alpha_i f(\mathbf{x}, \mathbf{a}_i),$$

where $f(\mathbf{x}, \mathbf{a})$ represents the entire class of regression tree functions; each $\mathbf{a}_i$ represents a specific set of joint parameter values realizing a member of this function class and $\alpha_0, \cdots, \alpha_m$ are the combination coefficients .

The gradient boosting decision tree Friedman (2001) iteratively adds new regression trees that fit the negative gradient of the loss function with respect to $F(\mathbf{x})$ at the most up-to-date estimate. Suppose that at the m-th step, we have in our ensemble a set of base learners $\{f(\mathbf{x}; \mathbf{a}_i)\}_{i=0}^{m-1}$ each of which takes the form of a regression tree and the corresponding weights $\{\alpha_i\}_{i=0}^{m-1}$. Then, the current estimation for the response of the j-th sample is given by

$$F_{m-1}(\mathbf{x}_j) = \sum_{i=0}^{m-1} \alpha_i f(\mathbf{x}_j; \mathbf{a}_i).$$

The negative gradient of loss function with respect to $F(\mathbf{x}_j)$ at $F_{m-1}(\mathbf{x})$ is $\tilde{g}_j$ and

$$\begin{aligned} \tilde{g}_j &= -\left[\frac{\partial L\left(F(\mathbf{x}_j)|\tau, \mathbf{c}, \mathcal{S}_l, \mathcal{S}_u, X\right)}{\partial F(\mathbf{x}_j)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} \\ &= \begin{cases} \hat{y}_i & \text{if } \hat{y}_i F(\mathbf{x}_i) < \hat{c}_i; \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{5.8}$$

The new base learner (which, in our case, is a regression tree) parameterized by $\mathbf{a}_m$ to be added to the ensemble at the current step is typically obtained by solving the following optimization problem:

$$\mathbf{a_m} = \arg\min \sum_{i=1}^{N} \left(\tilde{g}_i - f(\mathbf{x}_i; \mathbf{a}_m)\right)^2.$$

For simplicity, let us denote $f(x_i, a_m)$ by $f_m(x_i)$. Once $\mathbf{a}_m$ is fixed, the optimal line

search step size for $f_m(\mathbf{x})$ is obtained via

$$\begin{aligned}
& \arg\min_{\rho>0} L_1\left(F(\mathbf{x})+\rho f_m(\mathbf{x})|\tau,\mathbf{c},\mathcal{S}_l,\mathcal{S}_u,X\right) \\
= & \arg\min_{\rho>0}\sum_{i=1}^{N}\left[\hat{c}_i-\hat{y}_i F_{m-1}(\mathbf{x}_i)-\hat{y}_i\rho f_m(\mathbf{x}_i)\right]_+ \\
= & \arg\min_{\rho>0}\sum_{\hat{y}f_m(\mathbf{x}_i)>0}\hat{y}_i f_m(\mathbf{x}_i)\left[\frac{\hat{c}_i-\hat{y}_i F_{m-1}(\mathbf{x}_i)}{y_i f_m(\mathbf{x}_i)}-\rho\right]_+ \\
& -\sum_{\hat{y}f_m(\mathbf{x}_i)<0}\hat{y}_i f_m(\mathbf{x}_i)\left[\rho-\frac{\hat{c}_i-\hat{y}_i F_{m-1}(\mathbf{x}_i)}{y_i f_m(\mathbf{x}_i)}\right]_+ .
\end{aligned} \tag{5.9}$$

Let

$$r_i=\left[\left(\hat{c}_i-\hat{y}_i F_{m-1}(\mathbf{x}_i)\right)/\left(y_i f_m(\mathbf{x}_i)\right)\right], i\in\{1,\cdots,N, f_m(\mathbf{x}_i)\neq 0\}.$$

Since (5.9) is a piece-wise linear function, it is straightforward that the optimal ρ is one of r_i's that satisfies $r_i>0$, and could make (5.9) reach minimum.

When the number of instances N is very large, it is very time consuming to evaluate the function value for each r_i since each time of evaluation of the function value involves summing over N elements. However, since (5.9) is the difference of two monotonically decreasing functions, this repeated computation can be avoided by keeping track of the amount by which each function decreases each time when we increase ρ by a certain amount.

The description of the algorithm for solving censored regression with the truncated l_1 loss based on the gradient boosting approach is shown in Algorithm 6. The time complexity of the algorithm is $O(pMNlogN)$, where p is the number of features, thus it may not be applicable for large-scale datasets. Next we introduce the linear model which can be applied to large-scale datasets.

Algorithm 6 Gradient boosting approach for censored regression with truncated l_1 loss

Input: $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathcal{S}_l$, $\mathcal{S}_u$, $\mathbf{c}$, $\tau \in (0,1)$, M.

Output: $F_m(\mathbf{x})$.

$F_0(\mathbf{x}) \leftarrow \operatorname{argmin}_{z \in \mathbb{R}} L(z|\tau, \mathbf{c}, \mathcal{S}_l, \mathcal{S}_u, X)$

for $m = 1, \cdots, M$ **do**

  `Compute negative gradient` $\{\tilde{g}_i\}_{i=1}^N$ `by (5.8)`

  $\mathbf{a}_m \leftarrow \arg\min \sum_{i=1}^N \left(\tilde{g}_i - f(\mathbf{x}_i; \mathbf{a}_m)\right)^2$

  $\rho_t \leftarrow \arg\min_{\rho>0} L\left(F_{m-1}(\mathbf{x}) + \rho f(\mathbf{x}; \mathbf{a}_m)|\tau, \mathbf{c}, \mathcal{S}_l, \mathcal{S}_u, X\right)$

  $F_m(\mathbf{x}) \leftarrow F_{m-1}(\mathbf{x}) + \rho_m f(\mathbf{x}; \mathbf{a}_m)$

end for

5.4 A Linear Model for Censored Regression with l_1 Truncated Loss

In this section, we consider the case where the hypothesis takes the form of a linear combination of the covariates. We formulate the problem as follows:

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} L\left(\mathbf{w}^T \mathbf{x}_i | \tau, \mathbf{c}, \mathcal{S}_l, \mathcal{S}_u, X\right) + \frac{1}{2}\lambda \|\mathbf{w}\|_2^2, \tag{5.10}$$

where $\lambda > 0$ is the regularization parameter. In the following, we simply denote the loss function for the sample x_i as $L\left(\mathbf{w}^T \mathbf{x}_i\right)$. To take into consideration the effects of bias, we can append ones at the end of $\mathbf{x}_i$'s.

By the definition of the loss function in (5.2), the above problem can be written as

$$\begin{aligned} \min_{\mathbf{w}} \frac{1}{N} \Bigg[\tau \sum_{i \in \mathcal{S}_l} \left(c_i - \mathbf{w}^T \mathbf{x}_i\right)_+ + (1-\tau) \sum_{i \in \mathcal{S}_u} \left(\mathbf{w}^T \mathbf{x}_i - c_i\right)_+ \Bigg] \\ + \frac{1}{2}\lambda \|\mathbf{w}\|_2^2. \end{aligned} \tag{5.11}$$

Using the definition of auxiliary variables y_i, $\hat{c}_i$ defined in (5.3), (5.5) and further defining $\hat{\mathbf{x}}_i$ as:

$$\hat{\mathbf{x}}_i = \begin{cases} \tau y_i \mathbf{x}_i & \text{if } i \in \mathcal{S}_l; \\ (1-\tau) y_i \mathbf{x}_i & \text{if } i \in \mathcal{S}_u; \end{cases} \tag{5.12}$$

we can transform (5.11) into

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{N} \left[\sum_{i \in \mathcal{S}_l} \tau \left[y_i \left(c_i - \mathbf{w}^T \mathbf{x}_i \right) \right]_+ \right. \\ &\left. + \sum_{i \in \mathcal{S}_u} (1-\tau) \left[y_i \left(c_i - \mathbf{w}^T \mathbf{x}_i \right) \right]_+ \right] + \frac{1}{2} \lambda \|\mathbf{w}\|_2^2 \\ =& \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} [\hat{c}_i - \mathbf{w}^T \hat{\mathbf{x}}_i]_+ + \frac{1}{2} \lambda \|\mathbf{w}\|_2^2. \end{aligned} \tag{5.13}$$

The loss function above looks similar to the hinge loss used in SVM. However, the fact that $\hat{c}_i$ can be both positive and negative makes the popular packages such as LIBSVM and LIBLINEAR not applicable to solve this problem. An alternative is to transform the above problem into its dual form and use CVX to solve the resulting quadratic programming problem. However it is generally very slow and does not scale to large-scale datasets. Next, we show how to solve the problem with stochastic dual coordinate ascent which has been proven to achieve a fast convergence rate and can handle large-scale datasets.

Define $\phi_i : \mathbb{R} \to \mathbb{R}$ as $\phi_i(z) = [\hat{c}_i - z]_+$. Its convex conjugate $\phi_i^* : \mathbb{R} \to \mathbb{R}$ Boyd and Vandenberghe (2004) is

$$\phi_i^*(u) \equiv \max_{z \in \mathbb{R}} zu - \phi_i(z). \tag{5.14}$$

By plugging the definition of $\phi_i(z)$ into (5.14), we have

$$\phi_i^*(u) = \begin{cases} \hat{c}_i u & \text{if } u \in [-1, 0]; \\ +\infty & \text{otherwise.} \end{cases} \tag{5.15}$$

For $\mathbf{w} \in \mathbb{R}^d$, the convex conjugate of the regularization term $g(\mathbf{w}) \equiv \frac{1}{2}\|\mathbf{w}\|_2^2$ is

$$g^*(\mathbf{v}) \equiv \max_{\mathbf{w}\in\mathbb{R}^d} \mathbf{w}^T\mathbf{v} - \frac{1}{2}\|\mathbf{w}\|_2^2 = \frac{1}{2}\|\mathbf{v}\|_2^2.$$

The stochastic dual coordinate ascent (SDCA) Shalev-Shwartz and Zhang (2013) solves the dual problem which can be formulated as

$$\max D(\boldsymbol{\alpha})$$

where

$$\begin{aligned} D(\boldsymbol{\alpha}) =& \frac{1}{N}\sum_{i=1}^{N} -\phi_i^*(-\alpha_i) - \lambda g^*\left(\frac{1}{\lambda N}\sum_{i=1}^{N}\hat{\mathbf{x}}_i\alpha_i\right) \\ =& \frac{1}{N}\sum_{i=1}^{N}\hat{c}_i\alpha_i - \frac{\lambda}{2}\|\frac{1}{\lambda N}\sum_{i=1}^{N}\hat{\mathbf{x}}_i\alpha_i\|_2^2 \\ s.t. \quad & \alpha_i \in [0,1], \quad i = 1, \cdots, N \end{aligned} \tag{5.16}$$

and

$$\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_N].$$

With $g(\cdot) = g^*(\cdot) = \frac{1}{2}\|\cdot\|_2^2$, we can define $\mathbf{w}(\boldsymbol{\alpha}) = \nabla g^*(\mathbf{v}(\boldsymbol{\alpha}))$ where $\mathbf{v}(\boldsymbol{\alpha}) = \frac{1}{\lambda N}\sum_{i=1}^{N}\hat{\mathbf{x}}_i\alpha_i$. It is known that $\mathbf{w}^* = \mathbf{w}(\boldsymbol{\alpha}^*)$ where $\mathbf{w}^*$ and $\boldsymbol{\alpha}^*$ are the primal and dual optimal solutions with both the loss function and the regularization term in the primal problem being convex.

The SDCA method in each iteration randomly chooses one $\alpha_j\,(1 \le j \le N)$ to update with the objective of increasing the dual function value as much as possible.

That is

$$\begin{aligned}\Delta\alpha_j = \underset{\Delta\alpha_j\in\mathbb{R}}{\operatorname{argmax}} & -\frac{1}{N}\phi_j^*(-(\alpha_j+\Delta\alpha_j)) \\ & -\lambda g^*\left(\mathbf{v}(\boldsymbol{\alpha})+\frac{1}{\lambda N}\hat{\mathbf{x}}_j\Delta\alpha_j\right) \\ = \underset{\Delta\alpha_j\in\mathbb{R}}{\operatorname{argmax}} & \frac{1}{N}\hat{c}_j\left(\alpha_j+\Delta\alpha_j\right)-\frac{\lambda}{2}\|\mathbf{v}(\boldsymbol{\alpha})+\frac{1}{\lambda N}\hat{\mathbf{x}}_j\Delta\alpha_j\|_2^2 \\ s.t. \quad & -\alpha_j \le \Delta\alpha_j \le 1-\alpha_j. \end{aligned} \tag{5.17}$$

By expanding the l_2-norm and discarding the terms unrelated to $\Delta\alpha_j$, we can further get

$$\begin{aligned}\Delta\alpha_j = \underset{\Delta\alpha_j\in\mathbb{R}}{\operatorname{argmax}} & \ \hat{c}_j\left(\alpha_j+\Delta\alpha_j\right) \\ & -\frac{1}{2\lambda N}\|\hat{\mathbf{x}}_j\|_2^2\left(\Delta\alpha_j\right)^2-\mathbf{v}(\boldsymbol{\alpha})^T\hat{x}_j\Delta\alpha_j \\ s.t. \quad & \alpha_j+\Delta\alpha_j\in[0,1]. \end{aligned} \tag{5.18}$$

Then, by letting

$$t_j = \frac{\lambda N\left(\hat{c}_j-\mathbf{v}(\boldsymbol{\alpha})^T\hat{\mathbf{x}}_j\right)}{\|\hat{\mathbf{x}}_j\|_2^2}, \tag{5.19}$$

we have

$$\Delta\alpha_j = \begin{cases} -\alpha_j, & \text{if } t_j \le -\alpha_j; \\ t_j, & \text{if } -\alpha_j \le t_j \le 1-\alpha_j; \\ 1-\alpha_j, & \text{if } t_j \ge 1-\alpha_j. \end{cases} \tag{5.20}$$

The algorithm for solving (5.13) is shown in Algorithm 7. In practice, the number of iterations T can be determined by the duality gap which is the difference between the function value of (5.13) and (5.16) at the attained primal and dual solution. The iteration can be terminated when this gap drops below a predefined threshold.

Algorithm 7 Proposed SDCA algorithm for truncated loss based censor regression with linear model

Input: $X = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N\}$, S_l, S_u, $\mathbf{c}$, $\lambda \in \mathbb{R}^+$, T_0, T, $\tau \in (0, 1)$.

Output: $\bar{\mathbf{w}}, \bar{\boldsymbol{\alpha}}$

Initialize: $\boldsymbol{\alpha}^{(0)} = \mathbf{0}, \mathbf{v}^{(0)} = \mathbf{0}$

for $i = 1, \cdots, N$ **do**

  Compute $\hat{\mathbf{x}}_i, \hat{c}_i$ by (5.12), (5.5) respectively

end for

for $t = 1, \cdots, T$ **do**

  Randomly pick $j \in [1, N]$

  Compute $\Delta\alpha_j$ based on (5.20)

  $\boldsymbol{\alpha}^{(t)} \leftarrow \boldsymbol{\alpha}^{(t-1)} + \Delta\alpha_j \mathbf{e}_j$

  $\mathbf{v}^{(t)} \leftarrow \mathbf{v}^{(t-1)} + \frac{1}{\lambda N}\hat{\mathbf{x}}_j \Delta\alpha_j$

end for

$\bar{\boldsymbol{\alpha}} = \frac{1}{T-T_0} \sum_{i=T_0+1}^{T} \boldsymbol{\alpha}^{(t-1)}$

$\bar{\mathbf{w}} = \frac{1}{T-T_0} \sum_{i=T_0+1}^{T} \mathbf{v}^{(t-1)}$

5.5 A Multi-Stage Linear Model for Censored Regression

Based on the linear model introduced in the previous section, we propose in this section a mutli-stage linear model that can simultaneously estimate multiple linear models, one for each group of patients sharing the same number of treatment stages.

The key idea underlying our simultaneous estimation of models is that although patients may achieve remission from different stages of treatment, the stage-varying covariates collected on them around the same time before they entered follow-up study should share some commonalities in terms of their relative contribution to the prediction of relapse time and, at the same time covariates collected from different

stages overall, contribute differently to the prediction. In this section, by assuming that commonalities shared among the patients remitted across different treatment stages take the form of a linear combination of covariates, we show how these commonalities can be exploited toward simultaneous learning of the prediction model for multiple stages of treatment.

5.5.1 Formulation

Suppose that patients in the clinical study of interest experience at most M stages of treatment before they achieve remission. Let all the covariates of the ith patient who remitted after m treatment stages be $\mathbf{x}_i = \left[\mathbf{x}_{i0}^T, \mathbf{x}_{i1}^T, \cdots, \mathbf{x}_{im}^T\right]^T$ $(1 \leq m \leq M)$, where each of the $\mathbf{x}_{ik}^T$ $(0 \leq k \leq m)$ is a column vector and $\mathbf{x}_{i0}^T$ represents covariates that are not related to treatment such as those recording the patient's demographic information and family medical history information; $\mathbf{x}_k^T$ $(k \geq 1)$ represents the covariates from the treatment stage $k-1$ stages away from their last treatment stage. For instance, $\mathbf{x}_{i2}^T$ represents the covariates from second to the last treatment stage. Let $\mathbf{w} = \left[\mathbf{w}_0^T, \mathbf{w}_1^T, \cdots, \mathbf{w}_M^T\right]^T$ where $\mathbf{w}_k^T$ represents the coefficients associated with $\mathbf{x}_k^T$. Note that in our approach, the patients across all the treatment stages share the same weight vector as long as they have the covariates that the specific segment of weights corresponds to. Also let $\boldsymbol{\alpha} = [\alpha_{10}, \alpha_{11}, \cdots \alpha_{k0}, \cdots \alpha_{kk}, \cdots \alpha_{M0}, \cdots, \alpha_{MM}]$ be a vector of coefficients balancing the influence of different blocks of the covariates for patients that remit from each of the specific treatment stages. Let S_l^m and S_u^m be the set of indices of patients that have covariates from their last m treatment stages and have a lower bound and an upper bound in their relapse time, respectively.

Our proposed approach for simultaneously estimating censored regression models for patients remitting from multiple treatment stages can be formulated as follows:

$$\min_{\boldsymbol{\alpha},\mathbf{w}} \sum_{m=1}^{M} \left[\tau_m \sum_{i \in S_l^m} \left(l_i - \sum_{k=0}^{m} \alpha_{mk} \mathbf{w}_k^T \mathbf{x}_{ik} \right)_+ + (1-\tau_m) \sum_{i \in S_u^m} \left(\sum_{k=0}^{m} \alpha_{mk} \mathbf{w}_k^T \mathbf{x}_{ik} - u_i \right)_+ \right] + \lambda_1 \sum_{m=1}^{M} \sum_{k=0}^{m} \alpha_{mk}^2 + \lambda_2 \sum_{k=0}^{M} \|\mathbf{w}_k\|^2. \tag{5.21}$$

In making prediction, the coefficients for $\mathbf{x}_{ik}$ is $\alpha_{mk}\mathbf{w}_k$. All the covariates from the stage that is $k-1$ stages away from the last stage share the same $\mathbf{w}_k$ while for patients that experienced different number of stages, they have differ different α_{mk}.

5.5.2 Optimization

It is worth noting that although (5.21) is convex with respect to either $\boldsymbol{\alpha}$ or $\mathbf{w}$, it not jointly convex. The block coordinate descent algorithm that alternates between optimizing over $\boldsymbol{\alpha}$ and optimizing over $\mathbf{w}$ is adopted to solve this problem. In the following, we show that each step of the optimizing over $\mathbf{w}$ and optimizing over $\boldsymbol{\alpha}$ can be transformed into a problem of the same form as (5.11).

Fixing w, Solve $\boldsymbol{\alpha}$

When $\mathbf{w}$ is fixed, the problem (5.21) can actually be decoupled into M separate problems. Let $r_{ik} = \mathbf{w}_k^T \mathbf{x}_{ik}$. The problem (5.21) becomes M separate optimization problems, each of which takes the form:

$$\min_{\alpha_{m0},\cdots,\alpha_{mm}} \tau_m \sum_{i \in S_l^m} \left(l_i - \sum_{k=0}^{m} \alpha_{mk} r_{ik} \right)_+ + (1-\tau_m) \sum_{i \in S_u^m} \left(\sum_{k=0}^{m} \alpha_{mk} r_{ik} - u_i \right)_+ + \lambda_1 \sum_{k=0}^{m} \alpha_{mk}^2. \tag{5.22}$$

This problem is actually the same as (5.11), thus the algorithm introduced in the previous section can be used to solve it.

Fixing $\boldsymbol{\alpha}$, Solve w

When $\boldsymbol{\alpha}$ is fixed, the problem (5.21) turns into

$$\min_{\mathbf{w}} \sum_{m=1}^{M} \left[\sum_{i \in S_u^m} \left(\sum_{k=0}^{m} \mathbf{w}_k^T (\alpha_{mk}(1-\tau_m)\mathbf{x}_{ik}) - (1-\tau_m)u_i \right)_+ + \sum_{i \in S_l^m} \left(\tau_m l_i - \sum_{k=0}^{m} \mathbf{w}_k^T \left(\tau_m \alpha_{mk} \mathbf{x}_{ik}\right) \right)_+ \right] + \lambda_2 \|\mathbf{w}\|^2. \tag{5.23}$$

For $i \in S_u^m$, denote $\mathbf{p}_{ik} = \alpha_{mk}(1-\tau_m)\mathbf{x}_{ik}$; $\mathbf{p}_i = [\mathbf{p}_{i0}, \cdots, \mathbf{p}_{im}]$ and $\hat{u}_i = (1-\tau_m)u_i$. For $i \in S_l^m$, denote $\mathbf{q}_{ik} = \alpha_{mk}(1-\tau_m)\mathbf{x}_{ik}$; $\mathbf{q}_i = [\mathbf{q}_{i0}, \cdots, \mathbf{q}_{im}]$ and $\hat{l}_i = \tau_m l_i$. Let d_k be the dimension of $\mathbf{x}_{ik}$ and $\mathbf{I}_{d_k}$ be an identity matrix of d_k rows and columns. Also let $\mathbf{I}_M = \text{diag}[\mathbf{I}_{d_0}, \cdots \mathbf{I}_{d_M}]$ which is also an identity matrix and $\mathbf{I}_m$ be its first $d_0 + \cdots + d_m$ rows. Then the problem above can be simplified as

$$\begin{aligned} &\min_{\mathbf{w}} \sum_{m=1}^{M} \left[\sum_{i \in S_u^m} \left(\sum_{k=0}^{m} \mathbf{w}_k^T \mathbf{p}_{ik} - \hat{u}_i \right)_+ + \sum_{i \in S_l^m} \left(\hat{l}_i - \sum_{k=0}^{m} \mathbf{w}_k^T \mathbf{q}_{ik} \right)_+ \right] + \lambda_2 \|\mathbf{w}\|^2 \\ &= \min_{\mathbf{w}} \sum_{m=1}^{M} \left[\sum_{i \in S_u^m} \left(\mathbf{w}^T \mathbf{I}_m^T \mathbf{p}_i - \hat{u}_i \right)_+ + \sum_{i \in S_l^m} \left(\hat{l}_i - \mathbf{w}^T \mathbf{I}_m^T \mathbf{q}_i \right)_+ \right] + \lambda_2 \|\mathbf{w}\|^2. \end{aligned} \tag{5.24}$$

By introducing $\hat{\mathbf{p}}_i = \mathbf{I}_m^T \mathbf{p}_i$ and $\hat{\mathbf{q}}_i = \mathbf{I}_m^T \mathbf{q}_i$ and making use of the fact that $S_l = S_l^1 + \cdots + S_l^M$ and $S_u = S_u^1 + \cdots + S_u^M$, the above problem can be further written as

$$\min_{\mathbf{w}} \left[\sum_{i \in S_u} \left(\mathbf{w}^T \hat{\mathbf{p}}_i - \hat{u}_i \right)_+ + \sum_{i \in S_l} \left(\hat{l}_i - \mathbf{w}^T \hat{\mathbf{q}}_i \right)_+ \right] + \lambda_2 \|\mathbf{w}\|^2, \tag{5.25}$$

which also has the same form as (5.11).

Note that although in the STAR*D dataset, all the patients that achieved remission at a later stage have gone through all the previous treatment stages, this is not

a prerequisite for our model. Our model only requires that the covariates of all the patients are aligned in the reverse order of the treatment stages leading to remission.

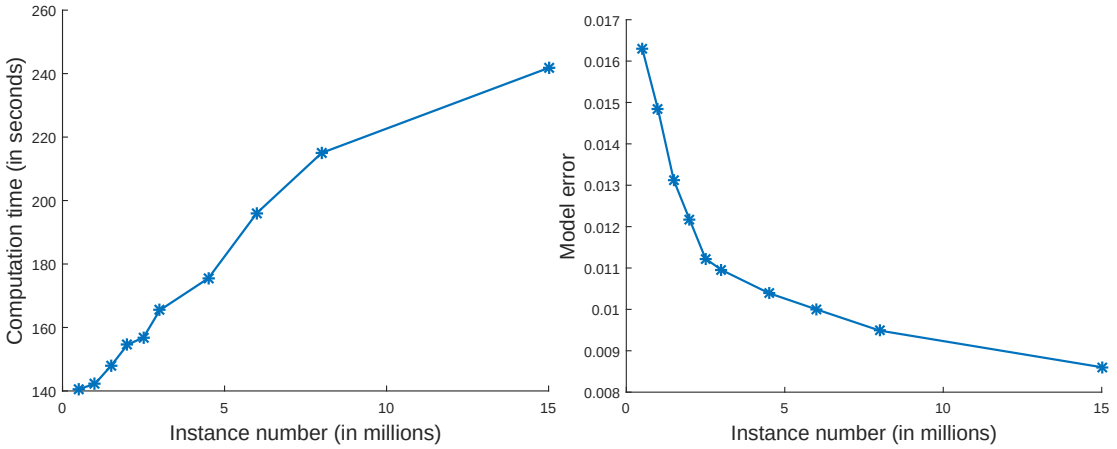
5.6 Experiments

5.6.1 Simulated Data

In this experiment, we use simulated data to answer the following two questions: (1) How well can our linear model scale to large datasets? (2) If there is an underlying linear relationship between the response and the covariates, to what extent that the linear model learned from minimizing the truncated l_1 loss recovers the true linear model when only an upper bound or a lower bound of the response of an instance is known?

Data Generation

We first generate our datasets as follows: Each instance is randomly drawn from a 100-dimensional standard normal distribution. For each dataset we generate n such instances where n ranges from 50k to 15,000k. The ground truth linear model $\mathbf{w}^*$ is a 100 dimensional vector with each entry being 0.5. The ground truth response for each instance is $y_i = \mathbf{w}^{*T}\mathbf{x}_i + e_i$ where $e_i \sim \mathcal{N}(0, 1)$; For each instance, we draw a random number b_i from $\mathcal{U}(0, \max_i |y|_i)$ and use $y_i + b_i$ as its upper bound or use $y_i - b_i$ as its lower bound. We assign a lower bound for half of the randomly chosen samples and an upper bound for the other half. In the simulation, we fix λ in (5.11) to be $1e-6$ and τ to be 0.5. The algorithm terminates when the duality gap is less than $1e-7$.



(a) Computational time

(b) Model error

Figure 5.1: Change of the computational time and model error when the number of instances increases.

Results

The simulation was run on a machine with Intel Xeon(R) CPU E5-1620 v2 3.70GHz $\times$ 8 processor, 32 GB memory and Ubuntu 14.04 LTS system. The results are shown in Figure 5.1. The computational time grows approximately linearly with the number of instances. The model error, measured by $\sqrt{\|\hat{\mathbf{w}} - \mathbf{w}^*\|_2^2/100}$, decreases as the number of instances increases. Note that as the number of instances increases, the perturbation range $\max_i |y|_i$ also increases, bringing in more uncertainty to the response of each instance. It is interesting to observe that even though only an upper bound or a lower bound of responses is provided, with enough samples, the linear model learned from minimizing the truncated l_1 loss can still to some extent recover the true model under certain conditions.

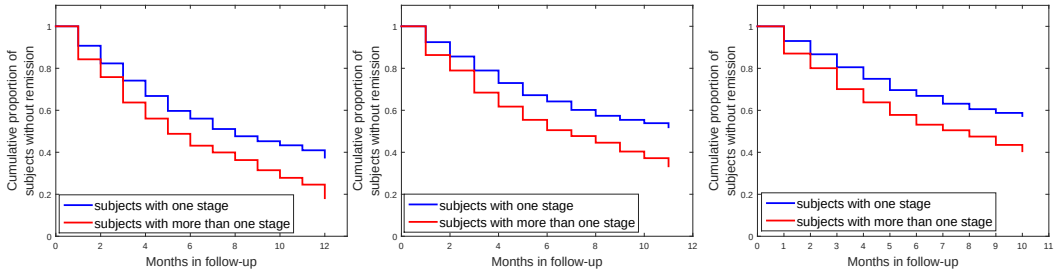
*5.6.2 Evaluation on STAR*D Cohort*

Dataset and Preprocessing

STAR*D is a study designed to identify the most effective treatment or combination of treatments for patients diagnosed with nonpyschotic MDD. It lasted over a period of seven years and involved over 4,000 patients aging from 18-75 and has been so far the largest and longest study ever conducted for evaluating the effectiveness of treatments of depression. It consists of four treatment stages during each of which patients were treated with certain antidepressants and their depressive symptoms were evaluated every two to three weeks. Patients that could not achieve remission or suffered from intolerable side effects of medications in one treatment stage were encouraged to proceed to the subsequent stage. Those who did achieve remission or demonstrated significant symptomatic improvement were invited to enter a 12-month naturalistic follow-up phase where assessments of patients' depressive symptom severity were made on a monthly basis.

Due to subjects dropping out without relapse in the follow-up phase, we considered in our experiments three cut-off points in the follow-up - 10 months, 11 months, 12 months respectively, in order to, on one hand, keep our analysis in as much accordance with the original design of the study as possible, one the other hand, take into account the subjects that dropped out at later time points of the follow-up phase and see how our model performs in response to changes in the total number of right censored cases. For each cut-off point, we included into our analysis only the subjects that either have definitively relapsed at or before the chosen cut-off time point (uncensored cases) and the subjects whose relapse occurred later than the chosen cut-off time point (right censored cases). It is worth emphasizing here that we excluded from our analysis

those who dropped out and did not relapse before the chosen cut-off time point so that risk of relapse for each sample is known.



(a) cut-off point at the 12th month (b) cut-off point at the 11th month (c) cut-off point at the 10th month

Figure 5.2: Kaplan-Meier survival curves of subjects in datasets determined by different cut-off points

As for the covariates, we included in our analysis those collected from the follow-up enrollment including demographics (DM), Eligibility (EL), Psychiatric Diagnostic Screening (PDS), etc. For each treatment stage, we took into consideration the covariates from patients' last observed record of Quick Inventory of Depressive Symptomatology - Clinician-rated (QIDS-C) and QIDS-SR (Self-rated), the baseline record of Interactive Voice Response (IVR) and Research Outcomes Assessments (ROA). According to Rush *et al.* (2006), relapse is defined as an individual having an observed QIDS total score collected in IVR during the follow-up phase great than 10. All the subjects included in our analysis achieved remission from the acute treatment stages. The number of subjects that relapsed and did not relapse by the cut-off time point from each treatment stage is shown in Table 5.1. Note that, for the cut-off time points before the 12th month, "non-relapse" cases included all subjects that had definitively not relapsed until that point, even if they eventually relapsed at some time later than that point. The Kaplan-Meier survival curves for subjects with only one treatment

Table 5.1: Data statistics of datasets determined by different cut-off points

	12th month			11th month			10th month		
Stage	Relapse	Non-relapse	Stage total	Relapse	Non-relapse	Stage total	Relapse	Non-relapse	Stage total
Stage 1	290	174	464	274	296	570	263	353	616
Stage 2	163	42	205	155	81	236	147	101	248
Stage 3	23	6	29	23	10	33	23	12	35
Stage 4	9	5	14	9	7	16	9	9	18
Total	485	227	712	461	394	855	442	475	917

stage and subjects with more than one treatment stage from different datasets are shown in Figure 5.2.

In training models, all the treatment stage varying covariates were aligned based on the reverse order of stages leading to remission. As is shown in Table 5.1 the number of subjects that remitted from stage 3 and stage 4 were too small, we omitted the covariates from the first treatment stage for those achieved remission from stage 3 and stage 4 as well as the covariates from the second treatment stage for those who achieved remission from stage 4. Missing values were imputed with the column mean. All the covariates that were included were normalized with zscore.

Feature Selection

The dataset included a large number of covariates, ranging from demographic information, medical and psychiatric comorbidities to depressive symptom measurements. However, not all of them are related to the subjects' relapse status at the end of the follow-up phase or risk of relapse, which necessitates feature selection as one of the crucial steps to minimize overfitting and ensure the quality of predictive models. In this work, l_1 sparse logistic regression-based stability selection Meinshausen and Bühlmann (2010) was employed to perform feature selection on each of the datasets determined by different cut-off points. The fitting target for sparse logistic regression is the relapse status by the cut-off point associated with each dataset. Since we have covariates from different stages of treatment, we ran stability selection twice, one on covariates from enrollment and last treatment stage, the other on covariates from second to last treatment stage. The number of covariates selected was determined by the cross-validation.

Performance Metrics

A commonly used metric for evaluating the performance of survival models is concordance index Steck *et al.* (2008) Khosla *et al.* (2010) which is a generalization of Area Under ROC Curve (AUC) for continuous response and censored data. It measures the probability of concordance between the predicted response and the observed response. A high concordance index means that there is a high likelihood that for two randomly sampled individuals, the order of their predicted response matches the order of their observed response. In our context, concordance index can be regarded as a measure of the proportion of the pairs of subjects for whom the relative order of predicted time of relapse is concordant with the order of observed time of relapse among all the pairs of subjects whose observed time of relapse can be ordered. Suppose we have n samples in our testing set. The observed time of relapse for the i-th subject is t_i. Its predicted time of relapse is p_i. Let $\mathcal{A}$ be the set of pairs that can be ordered, that is

$$\mathcal{A} = \{< i, j > | t_i > t_j, i, j = 1, \cdots, n\}.$$

Then the concordance index can be defined as

$$\text{Concordance Index} = \frac{1}{|\mathcal{A}|} \sum_{<i,j> \in \mathcal{A}} \boldsymbol{I}(p_i > p_j),$$

where $\boldsymbol{I}(\cdot)$ is the indicator function.

Experimental Setup and Results

We can divide each of the datasets into two sets of samples, one with data from one treatment stage only, the other with data from at least two treatment stages. For each dataset, we randomly selected 80% of samples from those who relapsed and those who did not relapse in each set as training set and the rest 20% as the testing set. A five fold cross-validation was carried out on the training set to select parameters.

This random split was repeated 10 times and the mean performance on the test sets and the standard deviation were reported.

The linear model, gradient boosting model, and Cox model were all trained and tested on the covariates from enrollment and last treatment stage. The model parameters including the number of features selected for all the models, τ, λ for linear models, τ, the number of trees in the gradient boosting model were determined by cross validation. For the multi-stage linear model, the number of covariates selected from different stages were determined independently and the τ's were set to be the same.

Table 5.2: Performance of different methods on dataset determined by cut-off at the 12th month

Methods	`Concordance Index`
Multi-Stage Linear	0.7172 (0.0200)
Linear	0.7003 (0.0267)
Gradient Boosting	0.6804 (0.0279)
Cox	0.6800 (0.0310)

Table 5.3: Performance of different methods on dataset determined by cut-off at the 11th month

Methods	`Concordance Index`
Multi-Stage Linear	0.7423 (0.0172)
Linear	0.7181 (0.0188)
Gradient Boosting	0.7020 (0.0238)
Cox	0.6952 (0.0257)

Table 5.4: Performance of different methods on dataset determined by cut-off at the 10th month

Methods	Concordance Index
Multi-Stage Linear	0.7443 (0.0110)
Linear	0.7242 (0.0164)
Gradient Boosting	0.7012 (0.0165)
Cox	0.6993 (0.0232)

The average concordance indices along with the standard deviation obtained by different methods on datasets determined by different cut-off time points are reported in Tables 5.2, 5.3, 5.4. Overall the performance on datasets with cut-off at the 10th and the 11th month is better, which is probably due to an increase in the number non-relapse cases that largely comes from subjects dropping out late in the follow-up phase. On all the datasets, our methods perform better than the Cox model. In particular, the multi-stage stage linear model produces the best performance, which, we think, can be accounted for by the fact that it can take into consideration the distributional difference of covariates from subjects remitted from different treatment stages. The performance of the gradient boosting approach is worse than that of the linear approach probably due to overfitting.

Identifying Risk Factors

One of the advantages of predicting relapse risk with linear models is that with all the covariates normalized to zero mean and the same variance, the magnitude of the coefficient associated with a covariate indicates its marginal contribution to the predicted risk, given all other covariates remaining unchanged. In this subsection,

we use the multi-stage linear models obtained from previous subsection based on random splits of the dataset determined by the cut-off point at the 10th month to produce a ranking of the covariates. Although the models built on different random splits of data involve different numbers of covariates, the numbers clustered in small range, with the number of covariates selected from Enrollment and last treatment stage varying from 70 to 90 and the number of covariates selected from the second to last treatment stage ranging from 10 to 20. Similar to the method used in Khosla *et al.* (2010) to cope with variance arising from cross-validation design, we averaged coefficients associated with each covariate over ten models and used the magnitude of the mean value minus their variance as a score to rank the coefficients.

Table 5.5 shows the top predictors of risk of relapse for subjects that experienced only one treatment stage. To our surprise, the "academic degree" comes on top of the list, suggesting that a higher academic degree is associated with a lower risk of relapse, according to the coding of this attribute and the sign of its coefficient. The residual symptoms, as mostly marked by "last observed", are also ranked high on the list, which corroborates the findings from Kennedy and Foy (2005) that the presence of residual symptoms such as depressed mood, hopelessness is associated with an earlier short-term relapse. Although residual sleep disturbance did not appear in our list of top predictors, which is consistent with the observation made in Nierenberg *et al.* (2010) that there is no significant different in Kaplan-Meier survival curves for those with and without the domain of residual sleep disturbance, we did find strong correlation between sleep onset insomnia at the baseline, which ranked second in our list of relapse risk factors. In addition, the subjective nature implied in some of the top-ranked predictors such as "Sad mood" and "impact of your family and friends" also help explain that mindfulness-based cognitive therapy Teasdale *et al.* (2000) can reduce the risk of relapse of MDD patients in remission or recovery.

The top predictors of risk of relapse for subjects that experienced more than one treatment stage largely overlap with those in Table 5.5, thus we did not show a separate table here. However, it is worth mentioning that the scores of covariates for patients with more than one treatment stage are generally lower and more flat and among top predictors, predictors marked with "*" in Table 5.5 rise to a much higher place in the list. But there are some predictors with a relatively high magnitude of mean in their corresponding coefficients but did not make into the top list due to a great variance, which probably results from relatively scarce presence of the conditions specified in those predictors among the subjects under study, implying that they should be considered as potential high risk factors if they are found in subjects. Such predictors include "Visited emergency room in last three months", "Careless work due to emotional problem" from baseline IVR of second to last treatment stage, "Tired nearly every day past 2 weeks" from PDS and "Family history drug abuse" from PHX. Overall, from the top risk factors we identified, we can see that therapies focusing on improving subjects' outlook for the future, psychomotor functioning and negative thinking might be more effective in preventing the relapse among the patients that achieved remission from treatment with antidepressant.

5.7 Conclusion

In this paper, we proposed a censored regression approach for predicting the risk of relapse of patients after their initial remission from one or multiple stages of antidepressant treatment. Since the patients' relapse status was assessed once every month, we employed a truncated l_1 loss to model the response for which only a lower bound or an upper bound is observed. We considered the hypothesis in the loss function that can be represented as (1) an ensemble of regression trees; (2) a linear combination of covariates. We developed an efficient gradient boosting algorithm when the hypothe-

Table 5.5: Top predictors from the multi-stage linear model for subjects with only one treatment stage

Predictor description	score	category
Academic degree	0.6630	Demographics
QIDS Sleep onset insomnia (baseline)	0.5074	IVR
Sum of QIDS sub-scores (baseline)	0.4327	IVR
American Indian or Alaskan Native	0.3481	Eligibility
QIDS Mood -sad (last observed)	0.3472	QIDS-SR
Impact of your family and friends	0.3436	Demographics
Worry obessively about you'd act/speak violently	0.3206	PDS
How often have you missed taking meds (last observed)	0.2962	QIDS-SR
Are you currently employed	0.2712	IVR
QIDS Concentration/decision making (last observed)	0.2621	QIDS-SR
Feel hopeless about future for 2 years	0.2615	PDS
QIDS total score (last observed)	0.2470	QIDS-SR
QIDS Psychomotor agitation (last observed)	0.2385	QIDS-C
Stomach and intestinal problems*	0.2349	PDS
Do impulsive things*	0.2311	PDS
QIDS Weight (decrease) last 2 weeks (last observed)	0.2109	QIDS-SR
IDS Psychomotor slowing (baseline)*	0.1944	RA
IDS Outlook-future (baseline)*	0.1850	RA
QIDS Concentration/decision making (baseline)	0.1841	IVR

Table 5.5: Top predictors from Multi-Stage linear model for subjects with only one treatment stage (cont.)

Predictor description	score	category
Number of answered QLESQ items	0.1719	IVR
Health been poor most of life*	0.1631	PDS
Depressed mood	0.1589	Eligibility
Worry when asking questions around others*	0.1588	PDS
Psychomotor agitation or retardation*	0.1543	Eligibility
On medical or psychiatric leave	0.1319	Demographics
Flashbacks of traumatic event	0.1215	PDS

sis takes the form of an ensemble of regression trees and a stochastic dual coordinate ascent algorithm when the hypothesis is a linear model. Furthermore, we extend the linear model to deal with covariates collected from multiple stages of treatment. Our experiments on synthetic data and STAR*D datasets demonstrate the efficiency and effectiveness of the proposed methods. In all cases, our multi-stage linear method achieves the best performance. In addition, the top risk factors identified by our multi-stage linear method are not only consistent with the findings from some of the recent research regarding relapse among patients with MDD who had initially achieved remission but also provided some insights into how to develop therapies for prevention of relapse.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Summary of Contributions

This dissertation proposes and studies a few methods to tackle some of the issues ubiquitous in data collected from patients with major depression and clinical trials on treatment of depression.

Specifically, in chapter 2, to deal with the presence of missing values and strong but possibly spurious empirical correlation between the variables due to relatively large number of variables and small number of samples, a method is proposed to learn a compressed set of representative features through an adapted version of sparse coding which is capable of simultaneously clustering variables with strong empirical correlation and dealing with the missing values in the design matrix. The method is tested on datasets of metabolic profiles and microarray gene expression profiles for the task of automatic diagnosis melancholic depression. In chapter 3, considering that the data collected from clinical trials of depression consists mostly of categorical variables and ordinal variables and decision tree based models are known for being capable of handling these types of variables, we propose a decision tree pruning method which is formulated into a non-convex optimization problem and overcomes the shortcomings of greedy nature and heavy reliance on heuristics of the traditional decision tree pruning approaches. Based on this method, we developed an approach to prune GBDT in chapter 4. This new approach eliminates the need to specify the number of trees and maximum depth of each tree and makes the training process more flexible and easily controllable with the addition of a single regularization parameter. The method is

tested on STAR*D for the task of prognosis (i.e. whether a patient is able to achieve symptom remission or meaningful treatment response by the end of the treatment stage(s)). Finally, in chapter 5, in our effort to tackle uncertainty in the actual time of relapse of which we are only sure of an interval that it falls in because of the monthly sampling of subjects' depressive syptom severity, we propose a formulation of censored regression with truncated l_1 loss and develop different methods to solve the resulting formulation.

It is worth noting that although the methods are proposed and evaluated in the context of a specific mental disorder - depression, they could be easily generalized to other biomedical applications as well since they are designed to handle broad categories of problems that are universally present in data like miccroarray gene expression profiles and clinical trial data, which are collected and analyzed in a wide variety of biomedical applications.

6.2 Future Work

Although the methods we propose and develop in the dissertation are able to achieve better performance in comparison to the baseline methods, they are without their limitations. As far as I am concerned, here are some of the aspects from which methods presented in this dissertation might be improved.

Robustness of sparse coding for variable clustering. The effectiveness of our method proposed in chapter 2 to generate representative features lies partly on how stable the solution given by sparse coding is. However, the solution of sparse coding as a non-convex optimization problem, is dependent upon the initialization and certain model parameters. Similar issues are also exhibited in most of current clustering techniques. Just as how consensus clustering addresses some of these issues with current clustering methods, a similar approach could be designed to improve the

stableness of solution of sparse coding which differs from the traditional clustering in that it allows different clusters to overlap.

Extension of decision tree pruning to multi-class classification. All the previous approaches of decision tree pruning is capable of dealing with multi-class classification. In chapter 3, the formulation we propose for decision tree pruning cannot directly handle multi-class classification problems. Although we can always convert multi-class classification problems into multiple one versus rest binary classification problems, it would be interesting to extend the method in such a way that it can take multi-class labels as input. Since most of the commonly used loss functions (e.g. least square loss, logistic loss) has been shown to be capable of working with multi-class problems, the key here lies in the development of algorithms to solve the proximal operator where the weight associated with each node is no longer a scalar but a vector instead.

BIBLIOGRAPHY

Alpert, J. and M. Fava, *Handbook of Chronic Depression: Diagnosis and Therapeutic Management*, Medical Psychiatry (Marcel Dekker Incorporated, 2014), URL `https://books.google.com/books?id=KFVPlzuOX88C`.

Amatriain, X., "Mining large streams of user data for personalized recommendations", ACM SIGKDD Explorations Newsletter **14**, 2, 37–48 (2013).

Appel, R., T. Fuchs, P. Dollár and P. Perona, "Quickly boosting decision trees-pruning underachieving features early", in "Proceedings of the 30th International Conference on Machine Learning, ICML 2013", pp. 594–602 (2013).

Asadi, N. and J. Lin, "Training efficient tree-based models for document ranking", in "Advances in Information Retrieval", pp. 146–157 (Springer, 2013).

Bifet, A., G. Holmes, B. Pfahringer and E. Frank, "Fast perceptron decision tree learning from evolving data streams", in "Pacific-Asia Conference on Knowledge Discovery and Data Mining", pp. 299–310 (Springer, 2010).

Boyd, S. and L. Vandenberghe, *Convex Optimization* (Cambridge University Press, New York, 2004).

Breiman, L., "Random forests", Machine Learning **45**, 1, 5–32 (2001).

Breiman, L., J. Friedman, C. J. Stone and R. A. Olshen, *Classification and regression trees* (CRC press, 1984).

Bühlmann, P., P. Rütimann, S. van de Geer and C.-H. Zhang, "Correlated variables in regression: Clustering and sparse estimation", Journal of Statistical Planning and Inference **143**, 11, 1835 – 1858 (2013).

Burgos-Artizzu, X. P., P. Dollár, D. Lin, D. J. Anderson and P. Perona, "Social behavior recognition in continuous video", in "2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", pp. 1322–1329 (IEEE, 2012).

Chen, T. and C. Guestrin, "Xgboost: A scalable tree boosting system", in "Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining- KDD'16", pp. 785–794 (ACM, 2016).

Cox, D. R., "Regression models and life-tables", Journal of the Royal Statistical Society. Series B (Methodological) **34**, 2, 187–220 (1972).

Dubey, R., J. Zhou, Y. Wang, P. M. Thompson and J. Ye, "Analysis of sampling techniques for imbalanced data an n=648 adni study", NeuroImage **87**, 0, 220 – 241 (2014).

Esposito, F., D. Malerba, G. Semeraro and J. Kay, "A comparative analysis of methods for pruning decision trees", IEEE Transactions on Pattern Analysis and Machine Intelligence, **19**, 5, 476–491 (1997).

Eyre, H. A., A. Eskin, S. F. Nelson, N. M. St. Cyr, P. Siddarth, B. T. Baune and H. Lavretsky, "Genomic predictors of remission to antidepressant treatment in geriatric depression using genome-wide expression analyses: a pilot study", International Journal of Geriatric Psychiatry (2015).

Freund, Y., R. Schapire and N. Abe, "A short introduction to boosting", Journal-Japanese Society For Artificial Intelligence **14**, 771-780, 1612 (1999).

Friedman, J. H., "Greedy function approximation: A gradient boosting machine", Annals of statistics pp. 1189–1232 (2001).

Friedman, J. H. and B. E. Popescu, "Predictive learning via rule ensembles", Ann. Appl. Stat. **2**, 3, 916–954 (2008).

Gabbay, V., R. G. Klein, Y. Katz, S. Mendoza, L. E. Guttman, C. M. Alonso, J. S. Babb, G. S. Hirsch, and L. Liebes, "The possible role of the kynurenine pathway in adolescent depression with melancholic features", J Child Psychol Psychiatry **51**, 935–943 (2010).

Gaynes, B. N., D. Warden, M. H. Trivedi, S. R. Wisniewski, M. Fava and A. J. Rush, "What did star*d teach us? results from a large-scale, practical, clinical trial for patients with depression", Psychiatric Services **60**, 11, 1439–1445 (2009).

Gong, P., C. Zhang, Z. Lu, J. Huang and J. Ye, "A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems", in "Proceedings of the 30th International Conference on Machine Learning, ICML 2013", pp. 37–45 (2013).

Hoheisel, J. D., "Microarray technology: beyond transcript profiling and genotype analysis", Nature reviews genetics **7**, 3, 200–210 (2006).

Johnson, R. and T. Zhang, "Learning nonlinear functions using regularized greedy forest", IEEE Transactions on Pattern Analysis and Machine Intelligence, **36**, 5, 942–954 (2014).

Kass, G. V., "An exploratory technique for investigating large quantities of categorical data", Applied statistics pp. 119–127 (1980).

Kelsey, J. E., "Achieving remission in major depressive disorder: The first step to long-term recovery", The Journal of the American Osteopathic Association **104**, S6–S10 (2004).

Kennedy, N. and K. Foy, "The impact of residual symptoms on outcome of major depression", Current psychiatry reports **7**, 6, 441–446 (2005).

Khosla, A., Y. Cao, C. C.-Y. Lin, H.-K. Chiu, J. Hu and H. Lee, "An integrated machine learning approach to stroke prediction", in "Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 183–192 (ACM, 2010).

Lee, C.-C., E. Mower, C. Busso, S. Lee and S. Narayanan, "Emotion recognition using a hierarchical binary decision tree approach", Speech Communication **53**, 9, 1162–1171 (2011).

Lee, J.-H., J.-H. Lee, S.-G. Sohn, J.-H. Ryu and T.-M. Chung, "Effective value of decision tree with kdd 99 intrusion detection datasets for intrusion detection system", in "Proceedings of the 10th International Conference on Advanced Communication Technology, 2008. ICACT 2008.", vol. 2, pp. 1170–1175 (IEEE, 2008).

Lichman, M., "UCI machine learning repository", URL `http://archive.ics.uci.edu/ml` (2013).

Lin, B., Q. Li, Q. Sun, M.-J. Lai, I. Davidson, W. Fan and J. Ye, "Stochastic coordinate coding and its application for drosophila gene expression pattern annotation", CoRR **abs/1407.8147** (2014).

Liu, J., S. Ji and J. Ye, *SLEP: Sparse Learning with Efficient Projections*, Arizona State University (2009).

Liu, J., L. Sun and J. Ye, "Projection onto a nonnegative max-heap", in "Advances in Neural Information Processing Systems", pp. 487–495 (2011).

Liu, Z., X. Li, N. Sun, Y. Xu, Y. Meng, C. Yang, Y. Wang and K. Zhang, "Microarray profiling and co-expression network analysis of circulating lncrnas and mrnas associated with major depressive disorder", PLOS One **9**, e93388 (2014).

Lu, X., H. Y. Yan, P. Yan, L. Li and X. Li, "Image denoising via improved sparse coding", in "British Machine Vision Conference", (BMVA Press, 2011).

Mairal, J., F. Bach, J. Ponce and G. Sapiro, "Online learning for matrix factorization and sparse coding", Journal of machine learning research **11**, 19–60 (2010).

Marcus, M., M. T. Yasamy, M. van Ommeren, D. Chisholm, S. Saxena *et al.*, "Depression: A global public health concern", WHO Department of Mental Health and Substance Abuse **1**, 6–8 (2012).

Mathias, M., R. Benenson, M. Pedersoli and L. Van Gool, "Face detection without bells and whistles", in "13th European Conference on Computer Vision (ECCV)-ECCV 2014", pp. 720–735 (Springer, 2014).

Mazure, C. M., M. B. B. Jr., F. H. Jr., K. B. Miller and J. Nelson, "Plasma catecholamine metabolites in subtypes of major depression", Biological Psychiatry **22**, 12, 1469 – 1472 (1987).

MB, K., L. PW, L. CE and K. GL, "Predictors of relapse in major depressive disorder", JAMA **250**, 24, 3299–3304 (1983).

MB, K., S. RW, L. PW and W. N, "Relapse in major depressive disorder: Analysis with the life table", Archives of General Psychiatry **39**, 8, 911–915 (1982).

McGrath, P. J., A. Y. Khan, M. H. Trivedi, J. W. Stewart, D. W. Morris, S. R. Wisniewski, S. Miyahara, A. A. Nierenberg, M. Fava and A. J. Rush, "Response to a selective serotonin reuptake inhibitor (citalopram) in major depressive disorder with melancholic features: A star*d report.", Journal of Clinical Psychiatry **69**, 12, 1847–1855 (2008).

Mehta, M., J. Rissanen and R. Agrawal, "Mdl-based decision tree pruning", in "Proceedings of the First ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'95", pp. 216–221 (1995).

Meinshausen, N. and P. Bühlmann, "Stability selection", Journal of the Royal Statistical Society: Series B (Statistical Methodology) **72**, 4, 417–473 (2010).

Mingers, J., "An empirical comparison of pruning methods for decision tree induction", Machine Learning **4**, 2, 227–243 (1989).

Niblett, T. and I. Bratko, "Learning decision rules in noisy domains", in "Proceedings of Expert Systems' 86, The 6Th Annual Technical Conference on Research and development in expert systems III", pp. 25–34 (Cambridge University Press, 1987).

Nierenberg, A. A., M. M. Husain, M. H. Trivedi, M. Fava, D. Warden, S. R. Wisniewski, S. Miyahara and A. J. Rush, "Residual symptoms after remission of major depressive disorder with citalopram and risk of relapse: a star*d report", Psychological Medicine **40**, 41–50 (2010).

Otte, C., "Incomplete remission in depression: role of psychiatric and somatic comorbidity", Dialogues Clin Neurosci **10**, 4, 453–460 (2008).

Parker, G. and D. Hadzi-Pavlovic, *Melancholia: A Disorder of Movement and Mood* (Cambridge University Press, New York, 1996).

Paykel, E. S., "Continuation and maintenance therapy in depression", British Medical Bulletin **57**, 1, 145–159 (2001).

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine learning in Python", Journal of Machine Learning Research **12**, 2825–2830 (2011).

Perlis, R. H., "A clinical risk stratification tool for predicting treatment resistance in major depressive disorder", Biological Psychiatry **74**, 1, 7 – 14 (2013).

Quinlan, J. R., "Induction of decision trees", Machine learning **1**, 1, 81–106 (1986).

Quinlan, J. R., "Simplifying decision trees", International journal of man-machine studies **27**, 3, 221–234 (1987).

Ramana, R., E. S. Paykel, Z. Cooper, H. Hayhurst, M. Saxty and P. G. Surtees, "Remission and relapse in major depression: a two-year prospective follow-up study", Psychological Medicine **25**, 1161–1170 (1995).

Rastogi, R. and K. Shim, "Public: A decision tree classifier that integrates building and pruning", in "VLDB", vol. 98, pp. 24–27 (1998).

Rush, A. J., M. H. Trivedi, S. R. Wisniewski, A. A. Nierenberg, J. W. Stewart, D. Warden, G. Niederehe, M. E. Thase, P. W. Lavori, B. D. Lebowitz, P. J. McGrath, J. F. Rosenbaum, H. A. Sackeim, D. J. Kupfer, J. Luther and M. Fava, "Acute and longer-term outcomes in depressed outpatients requiring one or several treatment steps: A star*d report", American Journal of Psychiatry **163**, 11, 1905–1917, pMID: 17074942 (2006).

Safavian, S. R. and D. Landgrebe, "A survey of decision tree classifier methodology", IEEE transactions on systems, man, and cybernetics **21**, 3, 660–674 (1991).

Segal, M. R., K. D. Dahlquist and B. R. Conklin, "Regression approaches for microarray data analysis.", Journal of Computational Biology **10**, 6, 961–980 (2003).

Shalev-Shwartz, S. and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization", Journal of Machine Learning Research **14**, Feb, 567–599 (2013).

Shivaswamy, P. K., W. Chu and M. Jansche, "A support vector approach to censored targets", in "Seventh IEEE International Conference on Data Mining, 2007. ICDM 2007.", pp. 655–660 (IEEE, 2007).

Steck, H., B. Krishnapuram, C. Dehing-oberije, P. Lambin and V. C. Raykar, "On ranking in survival analysis: Bounds on the concordance index", in "Advances in neural information processing systems", pp. 1209–1216 (2008).

Teasdale, J. D., Z. V. Segal, J. Mark, G. Williams, V. A. Ridgeway, J. M. Soulsby, M. A. Lau, J. D. Teasdale and V. A. Ridgeway, "Prevention of relapse/recurrence in major depression by mindfulness-based cognitive therapy", Journal of Consulting and Clinical Psychology pp. 615–623 (2000).

Thung, K.-H., C.-Y. Wee, P.-T. Yap and D. Shen, "Neurodegenerative disease diagnosis using incomplete multi-modality data via matrix shrinkage and completion", NeuroImage **91**, 0, 386 – 400 (2014).

Tibshirani, R., "Regression shrinkage and selection via the lasso", Journal of the Royal Statistical Society, Series B **58**, 267–288 (1994).

Tran, K., S. Hosseini, L. Xiao, T. Finley and M. Bilenko, "Scaling up stochastic dual coordinate ascent", in "Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", pp. 1185–1194 (ACM, 2015).

Xu, Z., G. Huang, K. Q. Weinberger and A. X. Zheng, "Gradient boosted feature selection", in "Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 522–531 (New York, NY, USA, 2014).

Yang, J., K. Yu, Y. Gong and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification", in "Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on", pp. 1794–1801 (IEEE, 2009).

Yu, C.-N., R. Greiner, H.-C. Lin and V. Baracos, "Learning patient-specific cancer survival distributions as a sequence of dependent regressors", in "Advances in Neural Information Processing Systems", pp. 1845–1853 (2011).

Zhang, T., B. Yu *et al.*, "Boosting with early stopping: Convergence and consistency", The Annals of Statistics **33**, 4, 1538–1579 (2005).